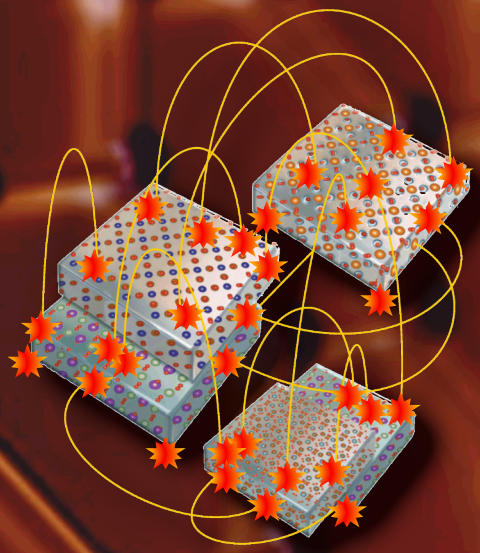
BLAISE PASCAL MAGAZINE 116

Multiplataforma /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js /
Bancos de dados / Estilos CSS / Aplicativos Web progressivos
Android / IOS / Mac / Windows & Linux



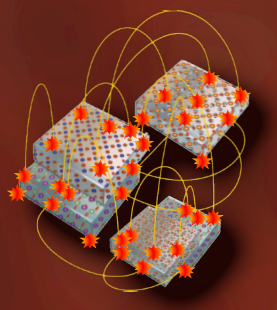
BLAISE PASCAL MAGAZINE 116

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

Blaise Pascal

CONTENT

ARTICLES



ADVERTISING

Niklaus Wirth

Pascal é uma linguagem de programação imperativa e procedural, projetada por Niklaus Wirth (esquerda abaixo) em 1968-69 e publicada em 1970, como uma linguagem pequena e eficiente destinada a incentivar boas práticas de práticas de programação usando programação estruturada e estruturação de dados. Um derivado conhecido como Object Pascal, projetada para programação orientada a objetos, foi desenvolvida em 1985. O nome da linguagem foi escolhido para homenagear o matemático, inventor da primeira calculadora: Blaise Pascal (veja o canto superior direito).

Editora: PRO PASCAL FOUNDATION em colaboração © Foundation Supporting Programming Language Pascal

CONTRIBUINTES

Stephen Ball
http://delphiaball.co.uk
DelphiABall

Dmitry Boyarintsev
dmitry.living @ gmail.com

Michaël Van Canneyt
,michael @ freepascal.org

Marco Cantù
www.marcocantu.com
marco.cantu @ gmail.com

David Dirkse
www.davdata.nl
mail: David @ davdata.nl

Benno Evers
b.evers @
everscustomtechnology.nl

Bruno Fierens
www.tmssoftware.com
bruno.fierens @ tmssoftware.com

Holger Flick
holger @ flixments.com

Mattias Gärtnernc-
gaertnma@netcologne.de

Max Kleiner
www.softwareschule.ch
max @ kleiner.com

John Kuiper
john_kuiper @ kpnmail.nl

Wagner R. Landgraf
wagner @ tmssoftware.com

Vsevolod Leonov
vsevolod.leonov@mail.ru

Andrea Magni
www.andreamagni.eu andrea.
magni @ gmail.com
www.andreamagni.eu/wp

Helmut Elsner
Korrektor der Deutschen
Ausgabe
helmut.elsner@live.com

Paul Nauta PLM Solution
Architect CyberNautics
paul.nauta @ cybernautics.nl

Kim Madsen
www.component4developers.com
kbmMW

Boian Mitov
mitov @ mitov.com

Jeremy North
jeremy.north @ gmail.com

Detlef Overbeek
- Editor in Chief
www.blaisepascal.eu
editor @ blaisepascal.eu

Anton Vogelaar
ajv @ vogelaar-electronics.com

Danny Wind
dwind @ delphicompany.nl

Jos Wegman
Corrector / Analyst

Siegfried Zuhr
siegfried @ zuhr.nl

Editor-chefe
Detlef D. Overbeek, Holanda Tel: Celular: +31 (0)6 21.23.62.68
Notícias e comunicados à imprensa somente por e-mail para editor@blaisepascal.eu

As assinaturas podem ser feitas on-line em www.blaisepascal.eu ou por ordem escrita, ou enviando um e-mail para office@blaisepascal.eu
As assinaturas podem ser iniciadas em qualquer data. Todas as edições publicadas no ano civil da assinatura também serão enviadas.
As assinaturas têm duração de 365 dias. As assinaturas não serão prorrogadas sem aviso prévio. O recibo de pagamento será enviado por e-mail.
As assinaturas podem ser pagas enviando o pagamento para: ABN AMRO Bank Account no. 44 19 60 863 ou por cartão de crédito ou PayPal
Nome: Pro Pascal Foundation (Stichting Ondersteuning Programeertaal Pascal)IBAN: NL82 ABNA 0441960863 BICABNANL2A VAT/NL814254147B01
Departamento de Assinaturas Edelstenenbaan 21 / 3402 XA IJsselstein, Holanda + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Member of the Royal Dutch Library KB Koninklijke Bibliotheek BIBLIOTECA REAL

Member and donor of WIKIPEDIA

SUBSCRIPTIONS 2024 prices)

	Shipment	TOTAL
Electronic Download Issue (8 per year) **±60 pages :**	**R$ 250**	**€ 348**

De seu editor

Olá, queridos leitores,
Assim como neste comentário, início expressando um alegre desejo para vocês: Feliz Páscoa!
As lebres não sairão de minha cartola, de acordo com um ditado holandês, mas certamente irão surpreendê-los.
Acredito que esta edição contenha muitas surpresas, algumas das quais são muito esperançosas:
Leiam o artigo sobre a Open Web criada na Europa. Há muita coisa acontecendo com ela e acredito que será um verdadeiro sucesso. Também há muito em jogo, é claro: a liberdade de expressão e a nossa democracia, às quais sou pessoalmente muito ligado, e a previsão de que a política terá que passar por um vale ainda mais profundo antes de perceber o que está fazendo.

Foi com grande tristeza que soube que Mattias Eissing, ainda tão jovem, faleceu. É uma pena que ele não tenha tido uma vida longa como a minha. Ele certamente teria feito muito pela comunidade. Desejamos força à sua família e aos seus amigos e enviamos nossas condolências.

Por outro lado (parece injusto), tenho 77 anos e espero continuar este trabalho por muito tempo. Acredito que não preciso convencer ninguém de que é muito trabalhoso fazer toda a revista. Ainda assim, espero obter uma ajuda extra de vocês, nossos leitores: precisamos muito de ajudantes práticos. Por exemplo, pessoas que tenham um bom domínio da língua inglesa ou portuguesa (brasileira). Isso é especialmente importante para manter a qualidade. Estamos precisando MUITO de leitores e desenvolvedores que queiram escrever artigos. Não é preciso ter medo. Escrever um artigo geralmente é muito divertido e ajuda a obter mais informações sobre seu próprio trabalho.

Na verdade, queremos criar nosso próprio fórum, onde nossos leitores possam interagir entre si. Para isso, é claro, precisamos de outro supervisor. Portanto, inscreva-se!
Para atrair os jovens, criei uma seção na qual pequenos trechos de código muito úteis podem ser exibidos. Também precisamos de envios para essa seção.
Além disso, gostaria de criar um conselho editorial, já que estou fazendo isso sozinho. Isso é importante para preparar a revista para o futuro e quero organizar minha própria divergência. Poderia haver uma divisão de trabalho. É claro que, como leitor, você também é bem-vindo para criticar.
Temos muito a ganhar com isso.

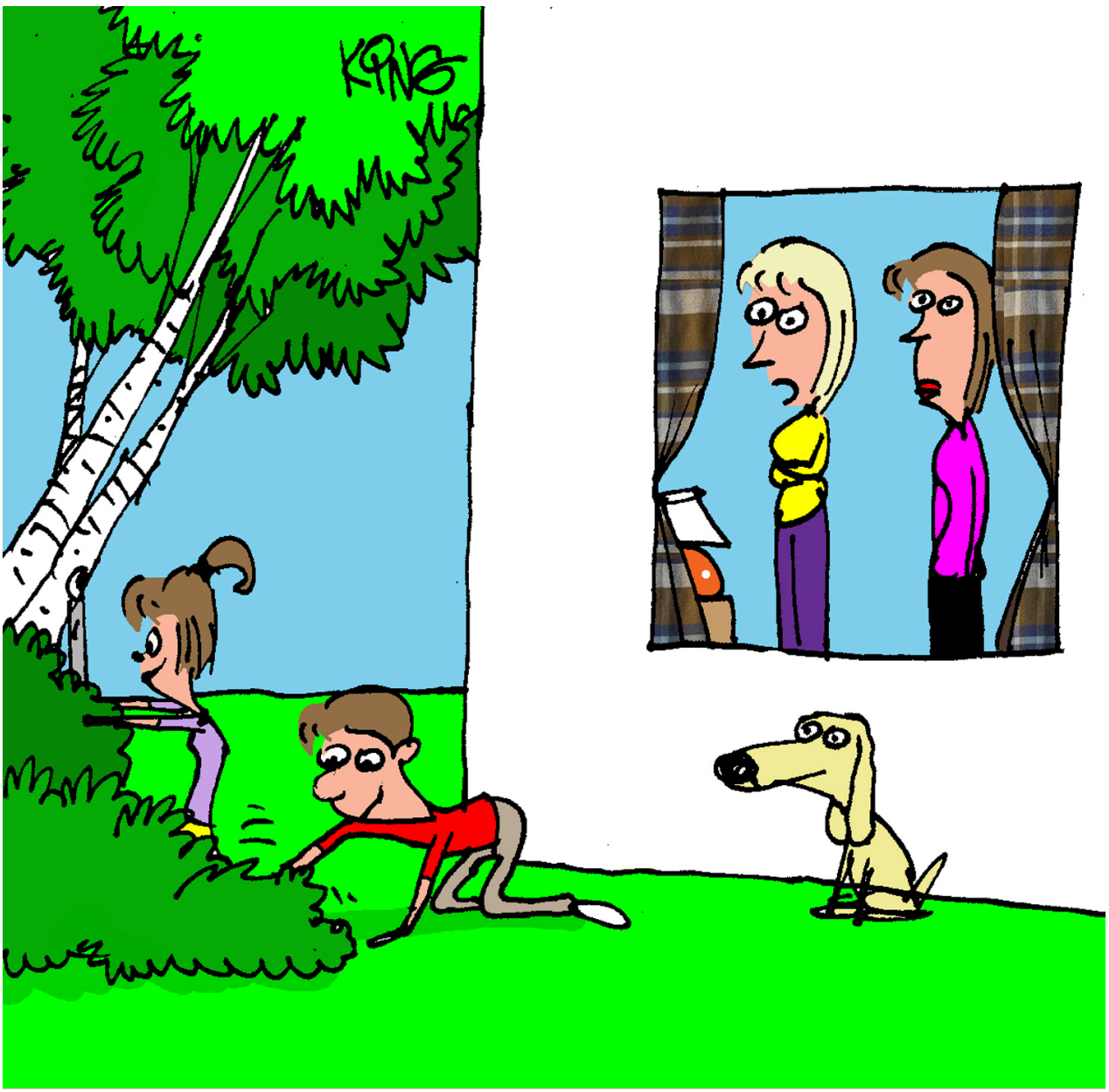
No momento, estou trabalhando com muita ênfase em uma primeira lição de Pascal para jovens. E isso começa de forma abrangente e no nível mais baixo, onde todos os conceitos abstratos são explicados de uma forma compreensível. Isso é OBRIGATÓRIO porque, como todos vocês sabem, precisamos de envolvimento. O melhor idioma merece os melhores jovens. Escreverei muito mais sobre isso...

Nesta edição, você também encontrará um anúncio sobre a Cúpula de 13/14 de junho. Nós também estaremos presentes com um estande. Talvez possamos nos conhecer melhor.

Atenciosamente,

Detlef
Seu editor

Do nosso consultor técnico, Jerry King



Eu tive que dizer a eles que havia telefones nos ovos de Páscoa. Essa é a única maneira de fazê-los sair e procurar os ovos que escondi.

GDK software

embarcadero
Official Technology Partner

Nós somos a GDK.

Tem um desafio Delphi e está à procura de experiência ou capacidade? Os nossos especialistas e programadores estão prontos para concretizar as suas ambições e objectivos.

30 **Deselvovedores Delphi**
Trabalhar com o nossos Experts em Delphi

99+ **Conversões Delphi**
Atualização inteligente para o Delphi mais recente.

5 **Embarcadero MVPs**
Autoridades na comunidade Delphi

4 **Escritórios pelo mundo**
Países Baixos, Reino Unido, Brasil e EUA

GDK EM POUCAS PALAVRAS

Sobre a GDK.

Partilhamos uma paixão pelo desenvolvimento de software e gostamos de nos manter a par das últimas tecnologias. Temos uma forte equipa de especialistas com muitos conhecimentos e experiência em Delphi.

ALCANÇANDO SUAS AMBIÇÕES

Trabalhar em conjunto.

Procura um parceiro para manter e aumentar o seu software? Ou atualizar o seu software para a versão mais recente do Delphi?
Nós estamos prontos para o ajudar!

Entre em contato

www.gdksoftware.com

SCAN ME

Codolex
Simplify development

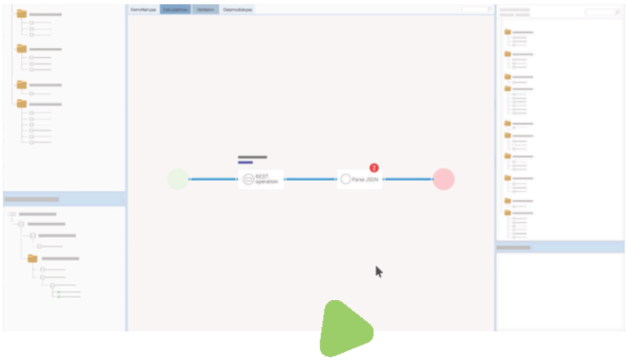
embarcadero
Official Technology Partner

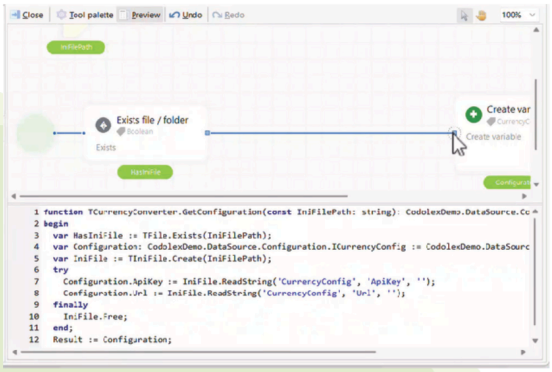
Revolucionário.

Codolex é uma solução de baixo código criada especificamente para o Delphi, que lhe permite desenvolver rapidamente, mantendo o controlo do código fonte.

Desenvolvimento visual

Desenvolva o seu código através do design visual. Compreenda a lógica mais rapidamente através da representação visual e ajuste-a facilmente modificando o fluxo. Uma imagem diz mais do que mil palavras!





Geração de código

A Codolex gera código que pode ser imediatamente utilizado nos seus projectos. Usando a função de pré-visualização no modelador, você pode ver imediatamente qual código é gerado para o fluxo. Não há dependência de fornecedor porque tudo é gerado em código.

A Codolex fornece tudo para simplificar o seu desenvolvimento

SCAN ME

www.codolex.com

Experimente

MATTHIAS EISSING †

É com pesar que lamentamos a perda de nosso estimado colega e amigo de longa data, Matthias Eissing, que nos deixou de forma repentina e inesperada em 14 de fevereiro de 2024, aos 53 anos de idade. Sua partida deixou a todos nós extremamente chocados e entristecidos. Matthias deixa sua companheira de muitos anos, Maria, e todos nós compartilhamos um profundo sentimento de luto por sua ausência.

Matthias, reconhecido por sua competência e habilidade técnica, deixa um vazio significativo entre seus colegas. Sua dedicação aos gatos, uma paixão que ele discretamente integrava em suas apresentações, era conhecida por apenas alguns. Como o principal apresentador em todos os eventos alemães, tanto presenciais quanto virtuais, Matthias era valorizado por sua expertise, habilidades e apoio, tanto pelos clientes quanto pelos colegas.

Como consultor de sistemas freelancer especializado em tecnologias da Borland Paradox, Matthias iniciou sua jornada com os produtos da Borland no início dos anos 90. A partir de 1997, sua dedicação o levou a se tornar um colaborador permanente, contribuindo com sua expertise para uma ampla gama de produtos de software da Borland, Inprise, CodeGear e Embarcadero. Entre todas as tecnologias, foi com o Delphi que ele estabeleceu uma conexão especial desde seu primeiro lançamento, demonstrando uma afinidade duradoura e profunda com essa plataforma.

Para nós, é muito comovente que Matthias tenha falecido em 14 de fevereiro, Dia dos Namorados, que também é o dia em que comemoramos o aniversário do Delphi, já que seu conhecimento do Delphi lhe rendeu o status de uma lenda na comunidade Delphi.

Desejamos a Maria e a todos os seus parentes muita força em seu luto.

Da equipe do Embarcadero Alemanha com texto adicional da equipe do Embarcadero EUA e Internacional.

Gostaríamos também de indicar a você o seguinte tópico no Delphi Praxis

Delphi Praxis / https://www.delphipraxis.net/214652-eine-sehr-traurige-nachricht.html

Notícias muito tristes

Alt 15 Fev 2024, 12:47

É com uma mistura de puro horror e grande tristeza que tenho que informá-los que nosso membro de longa data da comunidade, Matthias Eissing, faleceu repentina e inesperadamente na noite de quarta-feira.

Muitos de vocês conheciam Matthias não apenas por este fórum, é claro, mas também por ter tido relações profissionais com Matthias, já que ele foi consultor técnico da Embarcadero desde os primeiros dias da Borland até hoje. Mas é claro que também o conhecemos pessoalmente de inúmeros congressos e eventos ao vivo como um excelente palestrante. Eu mesmo tive a honra de fazer algumas apresentações com ele, como palestrante júnior, por assim dizer, e foi um prazer todas as vezes. Sentiremos falta do entusiasmo técnico de Matthias, aliado à sua aparentemente inesgotável riqueza de conhecimentos sobre o Delphi. Pude aprender muito com ele, tanto profissional quanto pessoalmente. Mal posso listar todos os lugares onde Matthias deixará uma lacuna gigantesca.

Matthias também me apresentou ao meu atual empregador anos atrás, algo que nunca esqueci e nunca esquecerei.

Matthias, estou muito, muito triste por não termos mais a oportunidade de conversar sobre compras durante uma cerveja, desviar do assunto e falar sobre algo completamente diferente depois e ainda nos divertirmos fazendo isso. Faça uma boa viagem e descanse em paz. Foi maravilhoso ter conhecido você.

Em luto silencioso,

Daniel

Daniel R. Wolf

Eine sehr traurige Nachricht

Alt 15. Feb 2024, 12:47

Mit einer Mischung aus blankem Entsetzen und großer Trauer muss ich Euch leider mitteilen, dass unser langjähriges Community-Mitglied Matthias Eißing in der Nacht zu Mittwoch plötzlich und völlig unerwartet verstorben ist.

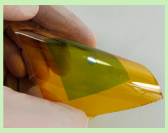
Viele von Euch kannten Matthias natürlich nicht nur hier aus diesem Forum, sondern werden auch beruflich mit Matthias zutun gehabt haben, da er als technischer Berater seit den frühen Borland-Zeiten bis heute bei Embarcadero intensiven Kundenkontakt pflegte. Aber natürlich kannten wir ihn auch persönlich von unzähligen Kongressen und Live-Veranstaltungen als exzellenten Referenten. Ich selbst durfte einige Vorträge mit ihm halten, als Junior-Referent quasi, und es war mir jedes einzelne Mal eine Freude. Matthias technische Begeisterung, gepaart mit seinem unerschöpflich wirkenden Delphi-Wissensschatz wird fehlen. Es gibt so vieles, was ich von ihm lernen konnte - fachlich, aber auch menschlich. Ich vermag kaum aufzuzählen, wo überall Matthias einer gigantische Lücke hinterlassen wird.

Matthias brachte mich vor Jahren auch mit meinem heutigen Arbeitgeber zusammen, was ich ihm nie vergessen habe und auch nie vergessen werde.

Matthias, ich bin sehr, sehr traurig, dass wir keine Gelegenheit mehr haben werden, bei einem Bier zu fachsimpeln, vom Thema abzudriften und hinterher über etwas ganz anderes zu sprechen und trotzdem Spaß dabei zu haben. Hab eine gute Reise und ruhe in Frieden. Es war wunderbar, Dich kennengelernt zu haben.

In stiller Trauer,
Daniel
Daniel R. Wolf

POR DETLEF OVERBEEK O artigo original foi publicado pelo MIT (Massachusetts Institute of Technology)



O **silício** é o segundo elemento mais comum no universo, depois do ar.
Ele compõe a maioria dos computadores que usamos atualmente. Formas de silício podem ser encontradas em rochas, argila, areia e sujeira. Embora não seja o melhor, é o mais fácil de obter para os eletrônicos.

Por isso, o silício é o material mais frequentemente encontrado em eletrônicos, como células solares, sensores e os chips dentro de computadores e smartphones.
Os engenheiros do **MIT** estão trabalhando em uma maneira de usar vários materiais diferentes, que não sejam de silício, **para produzir filmes muito finos que também sejam semicondutores**.

Os pesquisadores criaram filmes *dobráveis de arseneto de gálio, nitreto de gálio e fluoreto de lítio* para mostrar como seu método funcionava. Embora esses materiais funcionem melhor do que o silício, até agora eles eram muito caros para serem usados em produtos funcionais.
Os especialistas afirmam que o novo método torna fácil e barata a fabricação de componentes eletrônicos flexíveis a partir de qualquer mistura de partes semicondutoras. Esses circuitos funcionariam melhor do que os que usamos atualmente, que são baseados em silício.

Figura 1: nitreto de gálio (mineral)

Jeehwan Kim, que trabalha no **MIT** (*Massachusetts Institute of Technology - Massachusetts Avenue / Cambridge MA*) em Engenharia Mecânica e Ciência e Engenharia de Materiais, declarou,
"Descobrimos uma maneira de produzir eletrônicos flexíveis que podem ser fabricados com tantos tipos diferentes de materiais quanto o silício".
O MIT supõe que essa tecnologia possa ser usada para criar dispositivos de baixo custo que funcionem bem, como computadores portáteis e sensores, bem como células solares dobráveis.

TELA DE GALINHEIRO HEXAGONAL

Kim e seus colegas de trabalho criaram uma maneira de "copiar" materiais semicondutores caros com grafeno em 2017.

WIKIPEDIA

O grafeno é um alótropo* de carbono que consiste em uma única camada de átomos dispostos em uma nanoestrutura de rede hexagonal. O nome é derivado de "grafite" e do sufixo -ene, refletindo o fato de que o alótropo de carbono do grafite contém várias ligações duplas.

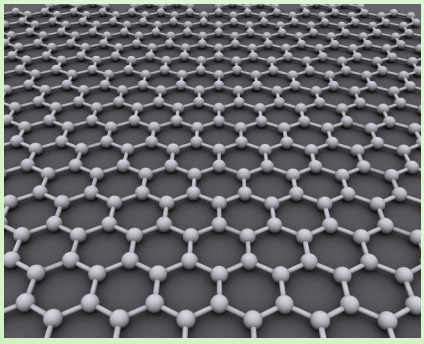
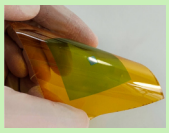


Figura 2: Fio de galinha hexagonal do grafeno

WIKIPEDIA

alotropia,
a existência de um elemento químico em duas ou mais formas, que podem diferir no arranjo dos átomos em sólidos cristalinos ou na ocorrência de moléculas que contêm diferentes números de átomos.



O **carbono** é capaz de formar muitos alotrópicos (*formas estruturalmente diferentes do mesmo elemento*) devido à sua **valência**. *A valência é o número de átomos de um determinado elemento que se combina com um átomo de outro elemento para formar uma molécula. A valência também é conhecida como peso molecular e é uma medida do poder de combinação de um átomo. A valência de um elemento é determinada pelo número de elétrons em sua camada mais externa*. Formas bem conhecidas de carbono incluem o diamante e o grafite.

Nas últimas décadas, muitos outros **alotrópicos** foram descobertos e pesquisados, incluindo formas esféricas como o **Buckminsterfullereno** (*é um sólido preto que se dissolve em solventes de hidrocarbonetos para produzir uma solução violeta*) e folhas como o grafeno.

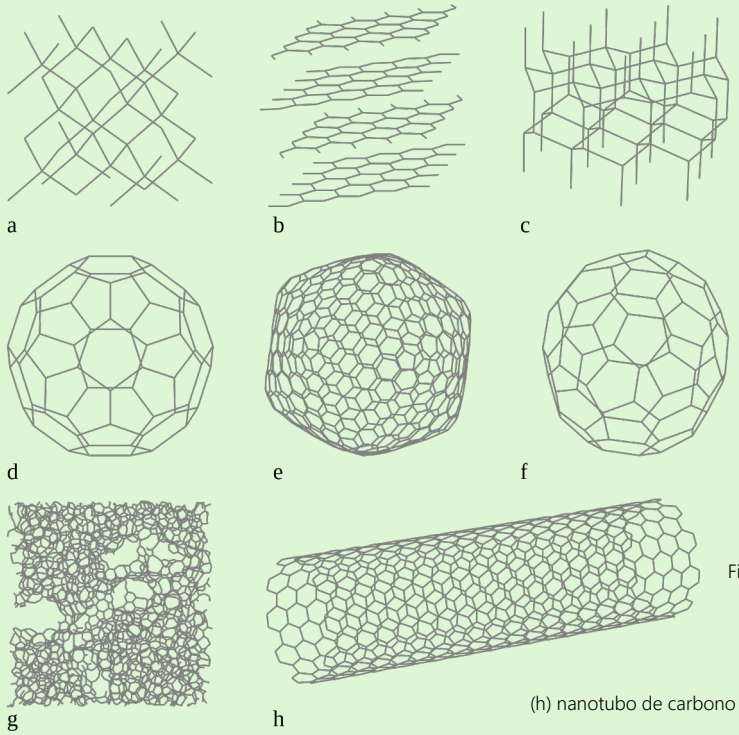


Figura 3: Oito **alótropos** de carbono:
(a) diamante,
(b) grafite,
(c) lonsdaleíta,
(d) C60 buckminsterfullerene,
(e) fulerita C540
(f) fulereno C70,
(g) carbono amorfo,
(h) nanotubo de carbono de parede simples em zigue-zague.

O **grafeno** é formado por uma camada muito fina de átomos de carbono agrupados de forma semelhante a uma tela de *galinheiro hexagonal*. Colocar o grafeno em um wafer puro e caro de um material semicondutor, como o arseneto de gálio, e permitir que os átomos de gálio e arseneto fluíssem pelo wafer fez parecer que os átomos estavam interagindo uns com os outros de uma forma que o grafeno entre eles parecia não estar presente. Assim, os átomos puderam se unir no padrão monocristalino exato do chip semicondutor abaixo. Isso permitiu fazer uma cópia perfeita que foi simples de remover a camada de grafeno. **O método econômico, chamado de "epitaxia remota (exposição tipo)"**, possibilitou a criação de muitas camadas de arseneto de gálio com apenas um chip caro embaixo.

A equipe pensou se o método poderia ser usado para fabricar outros materiais semicondutores logo após mostrar os primeiros resultados. Eles tentaram a epitaxia remota em silício e germânio, que são semicondutores baratos, mas quando colocaram os átomos sobre o grafeno, eles não se conectaram com as camadas abaixo. De repente, o grafeno deixou de ser transparente e passou a ser opaco, impossibilitando que os átomos de silício e germânio "vissem" os átomos do outro lado. O quartzo e o germânio fazem parte do mesmo grupo de elementos da tabela periódica. Para ser mais exato, esses dois elementos são ionicamente neutros, o que significa que não têm nenhuma polaridade. Eles estão na quarta classe de materiais. Isso mostrou ser uma dica.

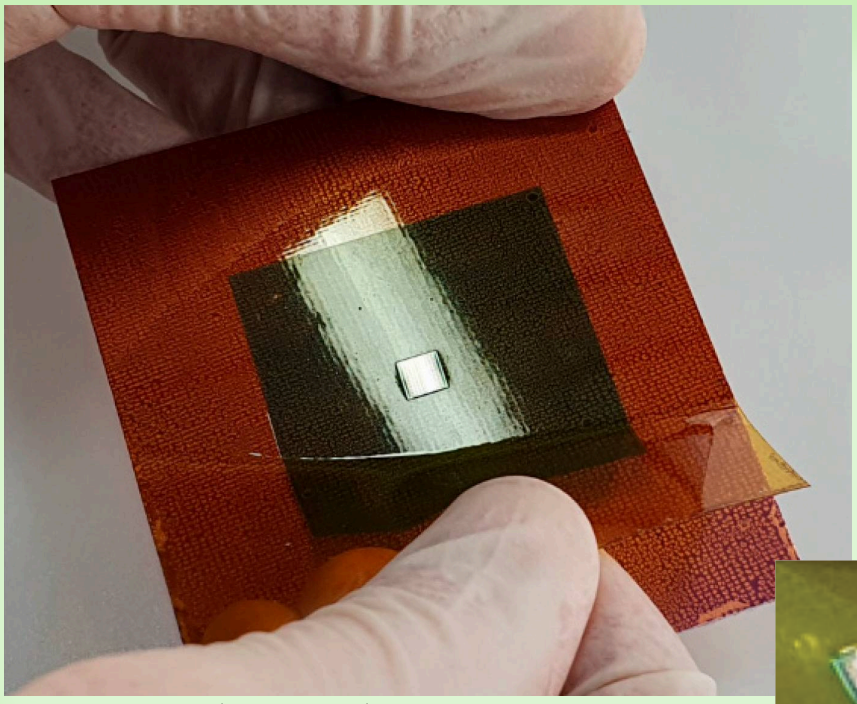


Figura 4: Descascamento do microprocessador

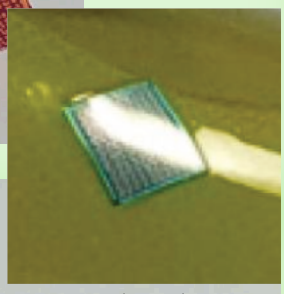


Figura 6: Ampliação da visão

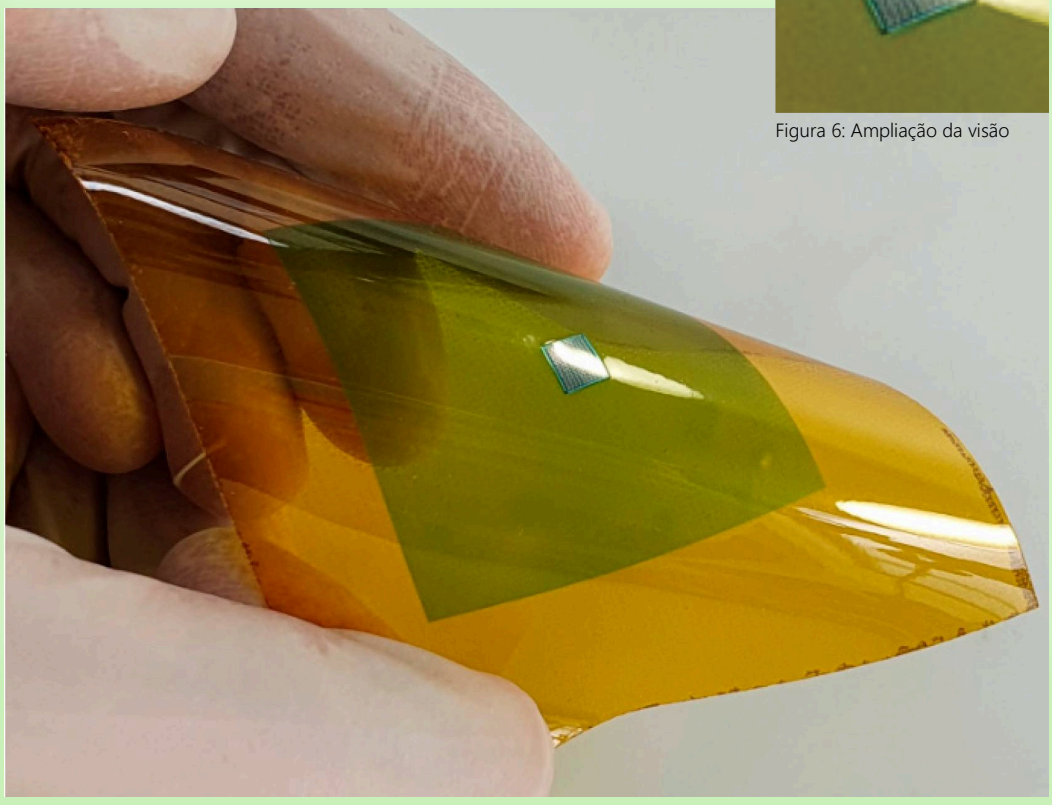


Figura 5: Observe a curvatura e a flexibilidade

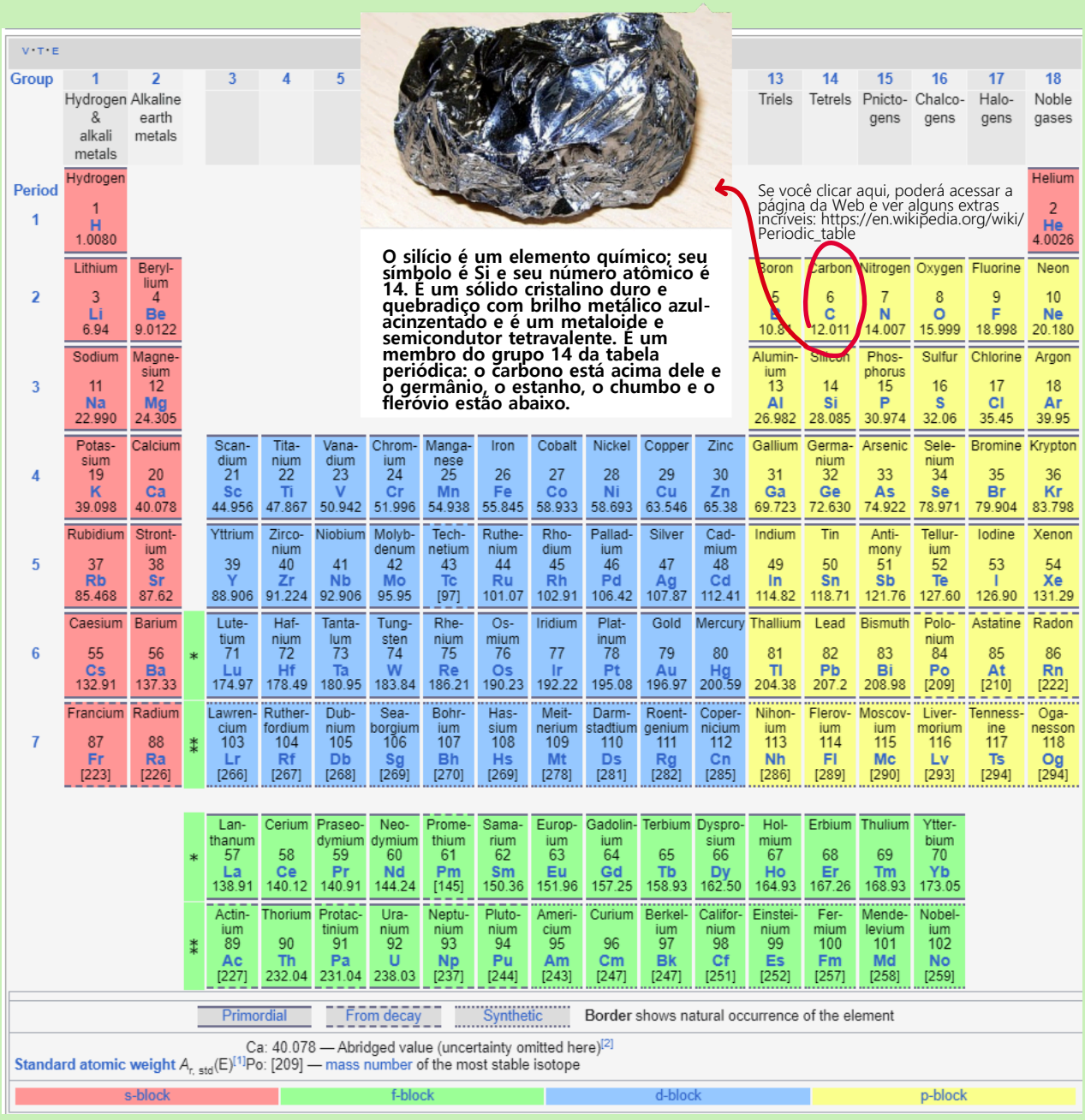
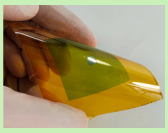


Figura 7: A tabela periódica. Se você clicar aqui, poderá acessar a página da Web e ver alguns extras incríveis



O grupo acreditava que o grafeno deveria ter sua própria carga iônica para que pudesse se combinar com outras moléculas. O arseneto de gálio, por outro lado, tem uma carga negativa no contato, enquanto o arsênio tem uma carga positiva.
Essa diferença de carga, ou polaridade, pode ajudar os átomos a repetirem o padrão básico de um átomo e a conversar entre si por meio do grafeno, como se ele fosse transparente.

O que eles descobriram foi que a maneira como o grafeno interage depende da polaridade dos átomos. *Em química, a polaridade é uma separação de carga elétrica que faz com que uma molécula ou seus grupos químicos tenham um momento de dipolo elétrico, com uma extremidade carregada negativamente e outra positivamente.* Até mesmo três camadas de grafeno podem interagir entre si para formar os materiais de ligação iônica mais fortes. O funcionamento é muito parecido com o de dois ímãs que podem se unir por meio de um pedaço de papel fino.

A equipe também descobriu que isso depende do material no qual os átomos interagem. Eles fizeram experimentos com o grafeno e com uma camada intermediária de nitreto de boro **hexagonal (hBN),** uma substância que imita o padrão atômico do grafeno e tem propriedades semelhantes às do **Teflon**, permitindo que os materiais sobrepostos se desprendam facilmente quando copiados.

Mas o **HBN** é feito de átomos de boro e nitrogênio com cargas opostas, o que lhe confere polaridade.
Em seus experimentos, os pesquisadores descobriram que todos os átomos que voavam sobre o **hBN**, mesmo que eles próprios fossem altamente polarizados, não conseguiam interagir totalmente com os wafers subjacentes.
Isso sugere que tanto a polaridade dos átomos principais quanto o material intermediário determinam se os átomos irão interagir e formar uma cópia da pastilha semicondutora original.
Portanto, parece que agora existem regras para a interação atômica do grafeno.
Os pesquisadores agora podem selecionar apenas dois elementos com cargas opostas usando esse novo entendimento para examinar a tabela periódica (veja a Figura 7 na página 4 deste artigo).
Eles podem fazer várias cópias precisas do wafer original usando métodos de epitaxia remota depois de fazer um wafer principal com os mesmos materiais.

Os wafers de silício são usados principalmente porque são baratos.
O uso de materiais que não sejam de silício agora é possível por meio desse método.
Você pode simplesmente comprar um wafer caro e copiá-lo de tempos em tempos, o que lhe permite reutilizar o wafer. A biblioteca de materiais para esse método está agora totalmente expandida.

Prevê-se que a **epitaxia remota** possa agora ser usada para produzir filmes ultrafinos e flexíveis a partir de uma ampla gama de materiais semicondutores anteriormente exóticos - desde que os materiais sejam feitos de átomos com uma determinada polaridade.
Mesmo em um futuro distante, esses filmes ultrafinos podem ser combinados para produzir dispositivos pequenos, flexíveis e versáteis, como sensores vestíveis, células solares flexíveis e até mesmo "telefones celulares que grudam na pele".
Kim diz: "**Precisamos de dispositivos de computação e sensores de baixa energia e alta sensibilidade**, feitos com materiais melhores, em cidades inteligentes, onde talvez queiramos colocar pequenos computadores em todos os lugares". "Este estudo abre as portas para esses dispositivos".

POCKET PACKAGE (2BOOKS) PRICE: € 40,00 EXCLUDING VAT AND SHIPPING

LAZARUS HANDBOOK

https://www.blaisepascalmagazine.eu/product-category/books/

Resolução de tela no Windows

Encontrei outra maneira de obter a resolução da tela no Windows que é consistente entre programas de 32 bits e 64 bits usando **GetSystemMetrics.**

```
uses
 Windows;

var
 ScreenWidth, ScreenHeight: Integer;

begin
 ScreenWidth  := GetSystemMetrics(SM_CXSCREEN);
 ScreenHeight := GetSystemMetrics(SM_CYSCREEN);

 WriteLn('Screen Width: ', ScreenWidth, ' pixels');
 WriteLn('Screen Height: ', ScreenHeight, ' pixels');
end.
```

Isso funciona bem e, com o **PTCGraph**, não há necessidade de se preocupar em descobrir quais modos de vídeo estão disponíveis, especialmente agora que podemos ter resoluções personalizadas para a janela do **PTCGraph**, só preciso garantir que ela caiba na tela, portanto, não importa como obtenho a resolução da tela.

James Richters

Como posso mostrar ao usuário que ele cometeu um erro?

Geralmente, é melhor fazer isso com uma exceção. Ela permite que você saia de maneira elegante de sua função ou procedimento. Isso acaba sendo muito útil se você criar dois procedimentos de erro padrão:
a) um que emite um bipe e sai da função/procedimento
b) um que também coloque um texto de erro na tela.

Exemplo: Use um procedimento com uma exceção Abort:

```
PROCEDURE AbortWithBeep;
BEGIN
  MessageBeep(-1); Abort;
END;
```

Use um procedimento com uma Exceção que coloque uma mensagem de erro na tela:

```
PROCEDURE ExceptOnMistake(Mistake : String);
BEGIN
  Messagebeep(-1);
RAISE Exception.Create('Error: '+ Mistake ;
END;
```

Como posso garantir que o usuário só possa inserir números?

Vamos supor por um momento que um campo Edit seja usado.
Então, você pode verificar a entrada em dois pontos:
a) ou quando o número inteiro for inserido,
b) ou assim que uma tecla for pressionada.

Por exemplo, se você quiser garantir que somente números (reais) sejam inseridos, crie o seguinte evento OnExit

```
PROCEDURE TForm1.Edit1Exit(Sender: TObject);
CONST LF=#13;{linefeed}
BEGIN
 IF Edit1.Text='' THEN
   BEGIN
     Edit1.SetFocus;
     ExceptOnError
       ('The amount is empty.'+LF+
        'We need to get the amount');
   END{if};

 TRY StrToFloat(Edit1.Text);
    EXCEPT
     Edit1.SetFocus;
     ExceptOnError
     ('The amount is not correctly typed.'+LF+
     'Type only a number, minus or comma');
    END{try};
END;
```

O Edit1 é verificado quanto a erros em 'onExit'.
O ExceptOnMistake descrito anteriormente retorna duas linhas de erro. A superior nomeia o erro e a inferior fornece a solução. Essa é uma boa prática. A constante LF (***avanço de linha***) é declarada localmente (***dentro do procedimento***) declarada.
Em seu programa, é melhor declará-la globalmente (no início do programa) para que possa ser usada em qualquer lugar. Se você quiser garantir que os inteiros sejam inseridos, use `StrToInt` em vez de `StrToFloat`.

Quando for óbvio qual erro foi cometido, um bipe será suficiente em vez de um texto de erro.
Nesse caso, use o `AbortWithBeep` mencionado anteriormente em seu procedimento:
O Edit1 agora é verificado a cada pressionamento de tecla com 'onKeyPress'. Se um valor tiver que ser digitado em Edit1, você poderá detectar uma letra digitada acidentalmente com:

```
PROCEDURE TForm1.Edit2KeyPress
(Sender: TObject; VAR Key: Char);
BEGIN
  IF NOT (Key IN [#8, ',', '-', '0'..'9'])
  THEN  AbortWithBeep;
END;
```

Key #8 garante que a tecla Backspace continue funcionando.
Não é necessário usar os procedimentos acima com um campo DBEdit. Quando o campo está vinculado a um campo número na tabela, a entrada é automaticamente validada.

NÃO É UM NÚMERO?
QUANDO UM NÚMERO NÃO É UM NÚMERO?
QUANDO É UM NAN.

POR DANNY WIND

DELPHI 12 AGORA USA NaN E INF

A partir do **Delphi 12**, um cálculo de ponto flutuante que anteriormente gerava uma exceção agora resultará em um valor **Not a Number (NaN) ou INF**.
No **Delphi 12**, todas as exceções de **ponto flutuante** agora são mascaradas por padrão. Nas versões anteriores do **Delphi**, três das exceções de FPU não eram mascaradas, e um cálculo poderia gerar uma exceção **Invalid Operation, Zero Divide** ou **Overflow** que precisava ser tratada no código com um try except. No **Delphi 12** para **VCL**, essas exceções não são mais geradas e agora obtemos valores de ponto flutuante **NaN** e **INF** como resultado de operação inválida e divisão por zero e cálculos de estouro.

FPU detection	Calculation	<= Delphi 11.3 (VCL)	>= Delphi 12 (VCL)
exInvalidOp	0.0 / 0.0 sqrt(-1)	EInvalidOp	NaN
exDenormalized	0.01 / 1e308	9,99999999999997E-311	9,99999999999997E-311
exZeroDivide	1.0 / 0.0	EZeroDivide	INF
exOverflow	1e308 + 1e308	EOverflow	INF
exUnderflow	1e-16 / 1e308	0.0	0.0
exPrecision	Most calculations	Closest round ties to even value	Closest round ties to even value

Observe que isso afeta principalmente os aplicativos VCL, pois os aplicativos FMX já usavam o novo modelo de exceção. No entanto, isso também afeta um pouco o FMX, pois o código subjacente agora também segue mais de perto o IEEE 754.

O QUE NÃO É UM NÚMERO

O conceito de NaN, ou Not a Number, em resposta a 0,0/0,0 ou sqrt(-1), pode ser surpreendente em um primeiro momento, mas logo começa a fazer sentido após uma reflexão cuidadosa.

Quando **William Kahan, o pai do ponto flutuante,** escreveu o livro de regras do **IEEE 754** para números de ponto flutuante ele o fez com cuidado e discernimento. Uma dessas percepções foi a inclusão do uso sistemático de um valor uso sistemático de um valor NaN, em que um resultado inválido de parte de um cálculo, como 0,0/0,0 poderia gerar um valor de ponto flutuante especial, um NaN, em vez de interromper o cálculo com uma exceção.
Dessa forma, todos os cálculos seguintes e adjacentes poderiam continuar e o **NaN** se tornaria parte desse cálculo. O NaN se propagaria para partes do resultado final, seguindo regras específicas. Dessa forma, os cálculos ainda poderiam produzir resultados parciais perfeitamente válidos, por exemplo, com matrizes, mesmo que algumas das etapas tivessem valores NaN. Isso funciona da mesma forma com os valores INFinity.

Na especificação **IEEE 754**, está escrito que você também pode lidar com essas e outras condições de limite, com exceções. Naquela época, o **Delphi** e várias outras linguagens de programação optaram por fazer exatamente isso. Mas, gradualmente, ao longo dos anos, cada vez mais linguagens de programação adotaram o uso de valores NaN e INF em vez de exceções.
Por quê?
Bem, isso resulta em cálculos mais rápidos e permite que os cálculos, especialmente as multiplicações de matrizes, falhem apenas parcialmente, ainda produzindo resultados corretos fora das condições de limite. Como agora a maioria das linguagens de programação adotou essa abordagem, chegou a hora do Delphi fazer o mesmo.

Com o novo **Delphi 12**, adotamos totalmente o NaN e o INF e eliminamos as exceções durante a execução de cálculos.
Entretanto, por causa disso, também temos que saber mais sobre esses valores especiais. E talvez também seja bom refrescar nosso conhecimento geral sobre ponto flutuante.

CONSIDERAÇÕES ESPECIAIS SOBRE NAN

Se considerarmos um NaN, ele é essencialmente um número desconhecido.
Isso significa que qualquer comparação com outro número ou mesmo com um NaN não faria sentido. Ele poderia ser tanto verdadeiro quanto falso. No IEEE 754, o resultado de uma comparação com NaN é sugerido para sempre resultar em um resultado Falso. Observe que esse comportamento não é obrigatório, cada linguagem e plataforma de CPU pode optar por implementar as comparações com NaN de forma diferente.
Nos cálculos, qualquer coisa que você calcular com um NaN deve resultar em um NaN também.
Ou, como apresentado no IEE 754: "Para uma operação com entradas NaN silenciosas, que não sejam operações de máximo e operações de máximo e mínimo, se um resultado de ponto flutuante tiver que ser entregue, o resultado deverá ser um
NaN que deve ser um dos NaNs de entrada. "

NaN <> x	True	unordered	NOT(NaN <> x)	False
NaN = x	False	means	NOT(NaN = x)	True
NaN > x	False	it	NOT(NaN > x)	True
NaN >= x	False	has	NOT(NaN >= x)	True
NaN < x	False	no	NOT(NaN < x)	True
NaN <= x	False	order	NOT(NaN <= x)	True

A interpretação das comparações de NaN não é ordenada, pois não é possível saber qual número NaN realmente representaria. A única comparação que seria verdadeira é que quando x é um número normal, ele não é igual a um NaN. Todas as outras comparações são falsas.

NAN RESULTADOS INESPERADOS

Essa definição não ordenada faz com que (NaN > x) seja avaliado como False, enquanto NOT(NaN <=x) é avaliado como True. No software, isso pode facilmente levar a resultados inesperados.
Por exemplo, no software de controle de velocidade de cruzeiro de um veículo. Se o código usar algo parecido com isso;

```
while cruise control enabled do
begin
  if NOT(MeasuredSpeed >= TargetSpeed) then
     {keep adding 0.1 to acceleration to get up to speed}
  else
     {reduce acceleration to -0.1 and stabilize}
end;
```

E se o projetista do medidor de velocidade decidisse retornar NaN se a medição for inválida, você logo estaria indo mais rápido do que pretendia.

DEBUGGING DE PROBLEMAS DE NAN E INF

É perfeitamente válido reativar as boas e velhas exceções da FPU para fins de debugging. Isso pode ser feito facilmente usando o seguinte código

```
uses System.Math;
    SetExceptionMask(exAllArithmeticExceptions-LegacyExceptionFlags);
```

Depois de reativar essas exceções, você pode lançar todos os tipos de testes unitários, de regressão e outros testes unitários, que você possa criar em seu software para eliminar todas essas exceções. Neste ponto, você deve estar se perguntando se não seria mais fácil deixar todo o código antigo como está e simplesmente reativar essas exceções legadas. Não é uma ideia ruim, mas ela tem algumas armadilhas. Uma delas é que o novo código base do **Delphi 12** segue mais de perto a norma IEE 754 em geral e você não pode voltar totalmente ao comportamento antigo.
Por exemplo, em comparações com NaN, o **Delphi 11.3** e o **Delphi 12** se comportam de forma diferente.

NaN > x	Platform	<= Delphi 11.3	>= Delphi 12
	VCL – Win32	True	False
	VCL – Win64	EInvalidOp	False
	FMX – Win32	True	False
	FMX – Win64	False	False

Os resultados que você obtém com o Delphi 12 são indiscutivelmente melhores e estão em conformidade com o IEEE 754. Você mesmo pode verificar isso com o Delphi 11.3 e o Delphi 12 com o seguinte trecho de código.

```
function NaN_larger_x: Boolean;
var
  lNaN, x: Double;
begin
  lNan    := Double.NaN;
  x       := 42.0;
  if (lNaN > x) then
    Result := True
  else
    Result := False;
end;
```

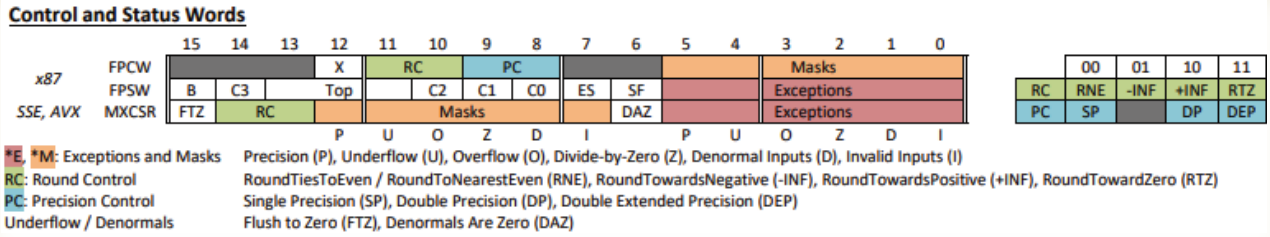
PALAVRA DE CONTROLE DA FPU

O padrão para a palavra de controle da **FPU 8087** também foi alterado no **Delphi 12.**
Esse valor só tem significado para **Win32 (x86 32-bits)** e influencia o arredondamento, a precisão e as exceções.

FPU Control Word	<= Delphi 11.3	>= Delphi 12
VCL – Win32	1372 0001 0011 0111 0010	Default 037F 0000 0011 0111 1111 Legacy 0372 0000 0011 0111 0010
FMX – Win32	137F 0001 0011 0111 1111	Default 037F 0000 0011 0111 1111 Legacy 0372 0000 0011 0111 0010

A coluna do **Delphi 12** lista o valor **8087 CW** quando todas as exceções da **FPU** estão mascaradas (Padrão) e quando as três exceções de **FPU** herdadas são desmascaradas (*Legado*).
Esperamos algumas dessas alterações, pois o (des)mascaramento de exceções faz parte da Control Word do 8087.
A máscara de exceção consiste nos últimos 6 bits à direita da CW do **8087** e esses valores são os esperados. Com todas as exceções mascaradas, esses bits são todos 1; para a configuração herdada, três deles são zero.

Mas o que está acontecendo com o quarto bit a partir da esquerda?
Por que esse bit agora é um zero em vez de um?



Na folha de referência de ponto flutuante da arquitetura Intel, aprendemos que o bit 12 é um bit reservado, X marca o local. Com um pouco de pesquisa no passado, parece que nas configurações das CPUs 80387 esse bit era reservado e usado **para habilitar o infinito positivo e negativo** em vez de apenas **o infinito único**. Outras fontes sugerem que ele pode ser usado para definir os valores normais a serem zerados ou salvos.
No entanto, o 8087 parece nunca ter suportado a descarga de valores normais para zero, no entanto, ele é compatível com o **SSE**.
Com base nessas informações, na especificação atual e em alguns testes, suspeito que esse bit não faz mais nada, e isso é apenas um pouco de limpeza.
Como observação adicional, o valor 037F é o valor padrão que a Intel usa para o 8087 CW.
Vou parar por aqui, embora haja muito mais a saber e aprender sobre números de ponto flutuante. Ainda acontece de o cálculo de **ponto flutuante** me surpreender e pode ser que você também se surpreenda de vez em quando.
Vou encerrar com este fato interessante:

FATO DIVERTIDO

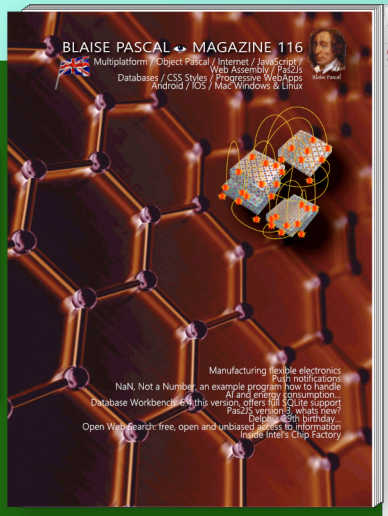
Você sabia que, se continuar adicionando 1,0 a um valor duplo, acabará obtendo 9.007.199.254.740.992 ?
Você pode pensar que esse é o valor máximo de um Double, mas não é.
É apenas o valor em que a precisão do Double se torna menor que 2, e o valor seguinte+1 é arredondado de volta para o valor mais próximo e exatamente igual.
É possível ir mais alto, mas é preciso adicionar mais do que apenas 1 para chegar ao próximo valor de ponto flutuante.
A etapa ou o intervalo entre os valores de ponto flutuante é conhecido como ULP, a Unidade no Último Lugar. Isso também significa que, após esse número, só há números pares.

LINKS E REFERÊNCIAS ÚTEIS (*basta clicar neles*)

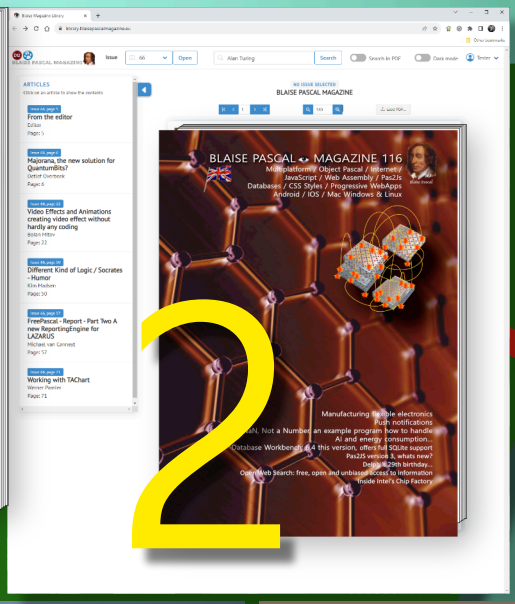
https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html

https://www.intel.com/content/www/us/en/developer/articles/technical/floating-pointreference-sheet-for-intel-architecture.html

ADVERTISEMENT



1



2

3

LAZARUS HANDBOOK

4

POCKET Edition +shipment

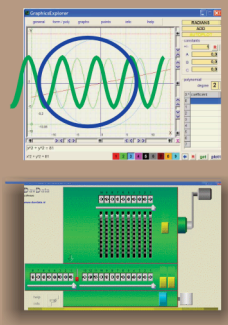
LAZARUS HANDBOOK PDF

5

6

LEARN TO PROGRAM USING LAZARUS
HOWARD PAGE-CLARK

DAVID DIRKSE
including 50 example projects



7

BLAISE PASCAL MAGAZINE

COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

1. One year Subscription
2. Internet Viewing of the Magazine
3. The newest LIB Stick
   - All issues 1-111
   - On Credit Card
4. Lazarus Handbook Pocket
5. LH PDF including Code
6. Book Learn To Program
   - using Lazarus PDF including 19 lessons and projects
7. Book Computer Graphics Math & Games
   - PDF including ±50 projects

SPRING OFFER

SUPER PACK

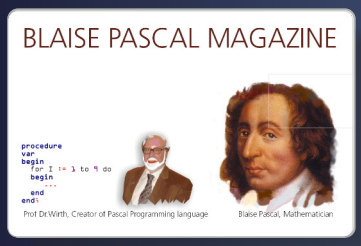
7 ITEMS

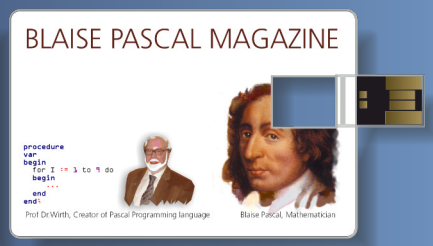
PRICE € 120

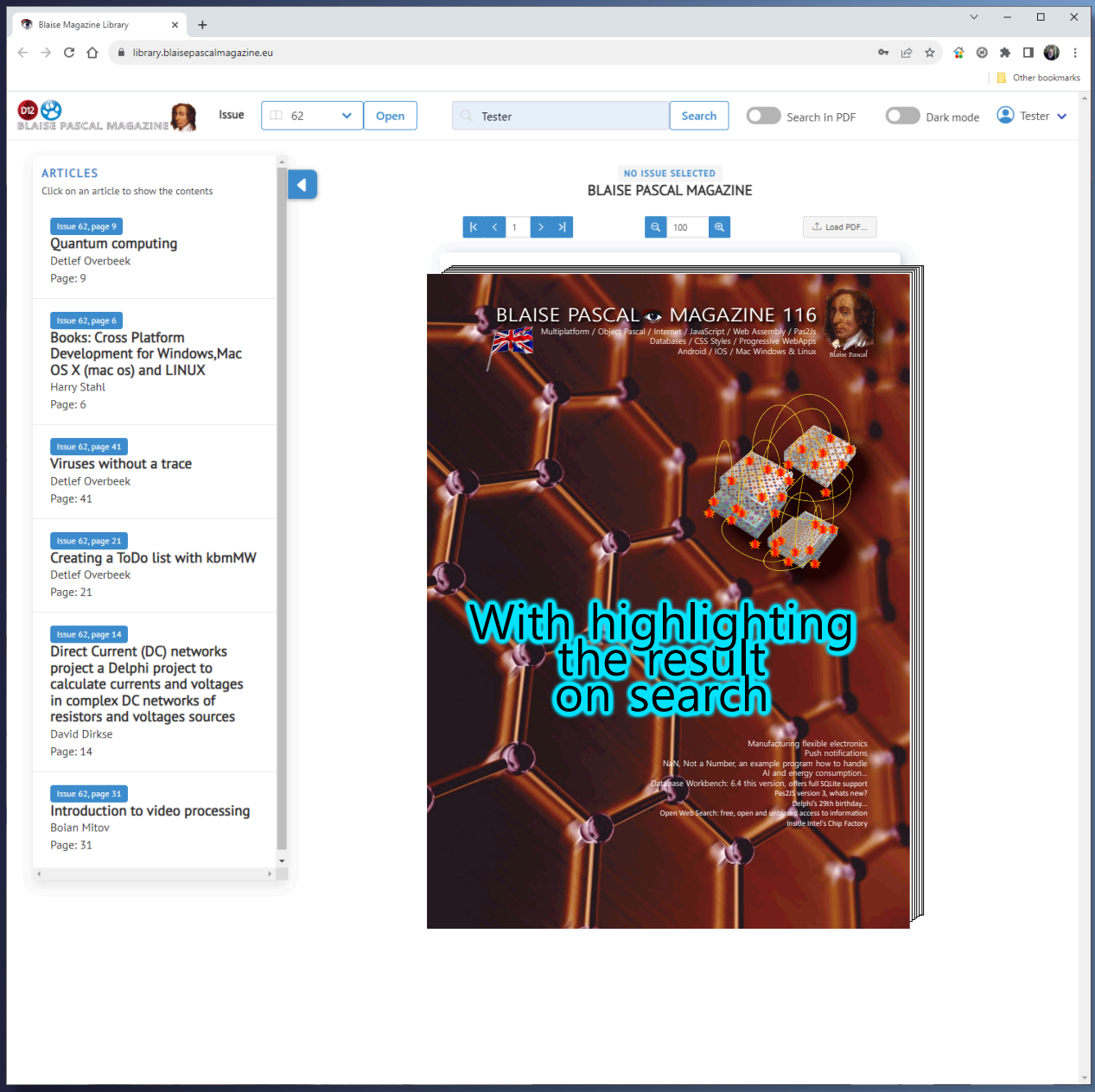
NORMAL PRICE € 275

LIB-STICK ON USB CREDIT CARD BLAISE PASCAL MAGAZINE

LIB-STICK USB-CARD: ALL ISSUES / CODE INCLUDED. SAME INTERFACE AS THE INTERNET LIBRARY € 100








BILLION
QUAL O
DESAFIO?

UM BILHÃO É LEGAL

POR IAN BARKER, DEFENSOR DOS DESENVOLVEDORES DA EMBARCADERO

Há uma cena fantástica no filme A Rede Social, durante o depoimento de Eduardo Saverin antes do julgamento, em que ele relembra o momento em que ele e Marc Zuckerberg encontram Sean Parker, fundador do Napster, pela primeira vez. Zuck, Eduardo e Sean estão discutindo a monetização do "The Facebook", como ainda era chamado na época. Na cena fictícia, depois que Marc e Eduardo sugerem que o site poderia valer uma quantia, Sean Parker profere uma frase imortal: "Sim, claro, você poderia fazer tudo isso e obter uma avaliação de um milhão de dólares. Mas você sabe o que é realmente legal? Um BILHÃO de dólares".

Porque um milhão não é suficiente
Esse é o ponto crucial das coisas. Atualmente, a sofisticação de nossa sociedade é tal que a palavra "milhão" é quase passé. Uma avaliação de um milhão de dólares em uma empresa dos EUA seria considerada quase minúscula. Enviar uma nave espacial em uma viagem de um milhão de milhas não é mais tão incrível assim. Já aconteceu, já foi feito. Isso também se aplica ao mundo da computação, onde ter um banco de dados com um milhão de registros é legal, mas, na verdade, há violações de dados que chegam regularmente a dezenas de milhões.

Parafraseando Sean Parker, sabe o que é legal? Um banco de dados com um BILHÃO de registros.

A ATRAÇÃO DO LADO SOMBRIO

Normalmente, não fico muito entusiasmado com desafios de codificação. Para mim, eles tendem a ser uma forma de onanismo intelectual. Muitas vezes, eles podem ser elitistas e ter uma espécie de clima de fraternidade do tipo "você não faz parte da nossa turma", o que impede a participação. Como a minha razão de ser é fazer com que o maior número possível de pessoas novas use o Delphi, eu me inclino a fazer coisas que levem as pessoas novas a verem o que o Object Pascal pode fazer por elas e mostrar por que eu acho que é a melhor escolha em quase todas as situações. As ações falam mais alto do que as palavras. Não vou conseguir atrair novos programadores desavisados para os encantos floridos das declarações begin/end e dos tipos fortes, fazendo trocas esotéricas sobre ciclos de CPU e conversas de nerd sobre magia negra reentrante, não é mesmo?

Mas, por outro lado, Object Pascal, a linguagem que todos nós amamos, também está viva, é capaz e pode dar um chute em muitos de seus pares em suas partes macias e moles quando se trata de velocidade, segurança de código e uma sublime sucinta expressão. Então, com isso em mente, você está disposto a flexionar seus músculos de codificação com raiva e realmente usar o Object Pascal como a arma de precisão que ele é? Você está? Então seja bem-vindo ao Billion Row Challenge.

O QUE É O DESAFIO DE UM BILHÃO DE LINHAS?

Ele surgiu graças ao MVP da Embarcadero, Gus Carreno. Veja como ele descreve as coisas:

Eu me deparei com um desafio muito interessante: Ler 1 bilhão de linhas de texto, processar alguns valores de temperatura, ordenar os valores agrupados e, em seguida, produzi-los.
Faça isso em um único thread, faça isso em vários threads, mas faça isso o mais rápido possível!

A ideia original veio do mundo Java: 1BRC em Java
`https://github.com/gunnarmorling/1brc`

Pensei imediatamente que esse seria um desafio maravilhoso e divertido para fazer em Object Pascal.

Se você acha que isso é algo de que gostaria, acesse o repositório 1BRC in Object Pascal, que pode ser encontrado aqui:
`https://github.com/gcarreno/1brc-ObjectPascal`

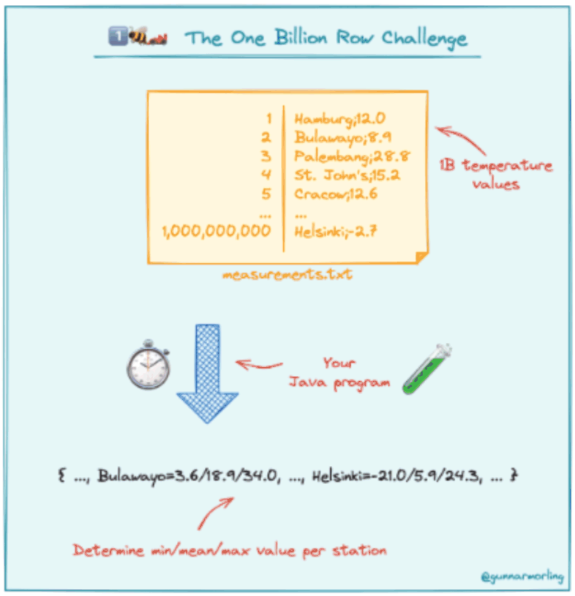
Os fundamentos desse desafio estão descritos lá, mas posso resumi-lo em uma única frase: Aprenda algo, ensine algo, mas, por favor, divirta-se fazendo isso.

The One Billion Row Challenge in Object Pascal

Delphi Community Discord 140 online | Unofficial Free Pacal Discord 82 online

Este é o repositório que coordenará o Desafio de 1 Bilhão de Linhas para Object Pascal.

O One Billion Row Challenge (1BRC) é uma exploração divertida de quão longe o Object Pascal moderno pode ser levado para agregar um bilhão de linhas de um arquivo de texto. Pegue todos os seus threads, use o SIMD ou qualquer outro truque e crie a implementação mais rápida para resolver essa tarefa!



O arquivo de texto contém valores de temperatura para uma série de estações meteorológicas. Cada linha é uma medida no formato <string: nome da estação>;<double: medida>, com o valor da medida tendo exatamente um dígito fracionário. O exemplo a seguir mostra dez linhas:

```
Hamburgo;12,0
Bulawayo;8,9
Palembang;38,8
São João;15,2
Cracóvia;12,6
Bridgetown;26,9
Istambul;6,2
Roseau;34,4
Conacri;31,2
Istambul;23,0
```

A tarefa é escrever um programa em Object Pascal que leia o arquivo, calcule o valor mínimo, médio e máximo da temperatura por estação meteorológica e emita os resultados em STDOUT da seguinte forma (ou seja, classificados em ordem alfabética pelo nome da estação e os valores do resultado por estação no formato <min>/<mean>/<max>, arredondados para um dígito fracionário, com o separador decimal sendo um ponto ., e para isso você pode escolher uma das opções apresentadas na Seção Arredondamento ou implementar a sua própria que seja consistente com as opções fornecidas):

```
{Abha=-23.0/18.0/59.2, Abidjan=-16.2/26.0/67.3, Abéché=-10.0/29.4/69.0, Accra=-10.1/26.4/66.4, Addis Ab
```

Como participar do desafio
Os envios serão feitos por meio de um PR (Pull Request) para este repositório.
O desafio será realizado de 10 de março a 10 de maio de 2024.
Ao criar sua inscrição, faça o seguinte:

1. Crie uma pasta em entradas com sua primeira inicial e sobrenome, por exemplo, para Gustavo Carreno: entradas/gcarreno.
2. Se estiver preocupado com o anonimato, porque a Internet é uma porcaria, sinta-se à vontade para usar um nome fictício: Bruce Wayne, Clark Kent, James Logan, Peter Parker, Diana de Themyscira. A escolha é sua!
3. Crie um README.md com algum conteúdo sobre sua abordagem, por exemplo, entries/gcarreno/README.md.
4. Coloque todo o seu código em entries/<seu nome>/src, por exemplo, entries/gcarreno/src.
5. Envie seu binário para a pasta bin na raiz desse repositório.
6. Se você precisar fornecer um .gitignore personalizado para algo que não esteja presente no principal, faça isso.
7. Leia o arquivo CONTRIBUTING.md para obter mais detalhes.

Esse desafio tem como principal objetivo nos permitir aprender algo novo. Isso significa que a cópia de código de outros será permitida, sob essas condições:

EXECUTING PROGRAMS ON THE SERVER IN PAS2JS

By Michael Van Canneyt

Starter → Expert

ABSTRACT

In this article we show how to give the user of a browser-based program feedback from long-running processes on the server, using 2 components: one in PAS2JS, one in Free Pascal/Lazarus.

❶ INTRODUCTION

When using a web-based program, not everything can be done in the browser.
Often,tasks are executed through some RPC (Remote Procedure Call) mechanism on the webserver. This can be a simple task such as executing an SQL statement on a database and returning a result, or it can be a more complicated and time-consuming task such as making a backup of a database, indexing PDF files, compiling software project and running a test suite, or even installing software on the server. Typically, the output of these remote programs should also be presented to the user.

To keep programs scalable, these tasks should be performed asynchronously: a timeout of 1 second for a HTTP request is already a long time, so executing a time-consuming task and waiting for the result using a single HTTP request is not a good idea:
the HTTP server is occupied with the request, the browser or any proxy servers between the HTTP server and the browser may decide to time-out your request.

Much better is to start the process using a HTTP request, and use a mechanism to poll the status of the executed process. In this article we present one such mechanism.

❷ ARCHITECTURE

The solution we present here consists of 2 components. One component which is used on the server, and which can be used to start a process, capture its output and poll for the status of the process. The other component takes care of the polling process on the client.

These components are ignorant of the communication mechanism between browser and server, this means that they do not implement the actual RPC calls used to start the process: There are many possible mechanisms, and some may be more suitable for your purpose than others.

The components are called `TProcessCapture` for the server part and `TProcessCapturePoller` for the client (PAS2JS) part. The server part takes care of executing a program and redirecting the output to a file, the client part implements the polling mechanism and some callbacks to handle the actual server calls and the result. We'll demonstrate both components with a simple set of programs:

- A test program to be executed.
  It is used for demonstration purposes only.
- A HTTP server program that allows to serve HTML files and that offers an
- RPC mechanism to start the test program and handle status requests. A Simple PAS2JS program that will run in the browser and which will remotely execute the test program. It will show the output of the test program in the browser.

We'll start with the test program.

With Higlight the result on search

RESUMO
Por um longo tempo, o **Free Pascal** não tinha suporte para **RTTI Extended**. Recentemente, o suporte a Extended RTTI foi incorporado à árvore de desenvolvimento principal, tornando o **Free Pascal** novamente mais compatível com o **Delphi**. Neste artigo, olharemos isso mais de perto.

❶ INTRODUÇÃO

Introspecção é a capacidade de examinar a estrutura dos dados (ou `Types`) em tempo de execução, sem saber exatamente de que tipo são os dados:
Ela permite que você examine o tipo usando alguma API fornecida pela linguagem de programação. No **Delphi** e no **Free Pascal**, o mecanismo de introspecção é chamado de **RTTI**: **Run-Time Type Information**.

Desde sua criação, o **Delphi** oferece suporte a uma forma limitada de **RTTI**:
As propriedades publicadas de uma classe podiam (*e ainda podem*) ser examinadas com o auxílio da **API RTTI:**
No início, a unidade `TypInfo`, depois a unidade `System.Rtti`. Esse mecanismo é usado para implementar o streaming, e o streaming é o que faz os arquivos de formulário funcionarem, tanto para o **FMX** quanto para o **VCL**.

O **Free Pascal** oferece suporte para essa forma de **RTTI** há muito tempo. O **Delphi 2010** estendeu o sistema **RTTI** não apenas das propriedades publicadas, mas também de todos os elementos de uma classe: campos, propriedades, e métodos poderiam ser examinados usando essa "**Extended RTTI**", não apenas para propriedades publicados, mas também para campos privados, protegidos e públicos.

Além disso, introduziu o `Attributes`:
um sistema para anotar a classe e seus campos, propriedades e métodos com informações adicionais.

❷ POR QUE RTTI ESTENDIDA?

Antes de nos aprofundarmos nos detalhes da **RTTI Api** e nas possibilidades, por que você gostaria de usar a **RTTI**? Um dos principais usos da **RTTI** é o **streaming**:
Ele permite que o senhor examine uma classe e grave o conteúdo dessa classe em um arquivo ou banco de dados.
Posteriormente, as informações no arquivo ou no banco de dados podem ser lidas e usadas para reconstruir o objeto.
Tudo isso pode ser feito por um sistema autônomo, que não precisa conhecer os detalhes das classes que grava ou lê: ele pode usar a **RTTI** para examinar as classes e manipular as classes.

Para tornar isso mais específico, veja a **class** `TTimer`:

```
TComponent= class (TPersistent)
published

Property Name : string Read FName Write SetName;
end;

TTimer = class (TCustomTimer)
published
  property Enabled: Boolean read FEnabled write SetEnabled default True;
  property Interval: Cardinal read FInterval write SetInterval default 1000;
  property OnTimer: TNotifyEvent read FOnTimer write SetOnTimer;
  property OnStartTimer: TNotifyEvent read FOnStartTimer write FOnStartTimer;
  property OnStopTimer: TNotifyEvent read FOnStopTimer write FOnStopTimer;
end;
```

Por meio da **RTTI**, um sistema de streaming sabe que a classe TTimer tem 6 propriedades que podem ser transmitidas. Ele sabe o tipo da propriedade (por exemplo, `cardinal` para **Interval**), como ela deve ser lida (diretamente do campo `FInterval`) ou gravada (por meio do **method** `SetInterval`), e qual é o valor padrão (1000):
O valor padrão é uma forma de otimizar a gravação, não escrevendo o valor de uma **property inspector** se ele corresponder ao valor padrão. O mesmo mecanismo permite que o inspetor de propriedades mostre objetos que ele não conhece.
A **RTTI** acima tinha algumas limitações:
Somente as **published properties** podem ser examinadas. O tipo de informação que poderia ser publicada também era limitado: classes descendentes de `TPersistent`, ordinal types, set types (até 32 elementos) e tipos de `float`. Os métodos também precisavam ser publicados para que pudessem ser usados como manipuladores de eventos.
Portanto, nada de registros ou matrizes, propriedades públicas ou protegidas ou simplesmente campos. A ausência de anotações significava também que o sistema de **streaming** não tinha a possibilidade de armazenar informações extras (*por exemplo, um nome alternativo para uma propriedade*).
Com a introdução do **Extended RTTI** e dos **Attributes**,, essas restrições foram eliminadas:
Informações de qualquer tipo podem ser examinadas, campos podem ser examinados.
Todas as visibilidades ((**`public, protected, private`**) agora estão sujeitas a inspeção.

❸ POR QUE ATRIBUTOS?

Para ver por que as anotações podem ser úteis, suponha que o senhor tenha um serviço **REST** externo que serve e consome dados por meio de **JSON**. Por exemplo, um objeto **JSON** de pessoa de contato

```
{
"first-name" : "michael",
"last-name" : "Van Canneyt"
"date-of-birth" : "19700707"
}
```

Se quiser modelar esses dados em um objeto, você precisará usar identificadores e tipos pascal:

```
TPerson   = record
firstname : string;
lastname  : string;
birthdate : TDateTime;
end;
```

Dois problemas se tornam imediatamente aparentes: os nomes de campo (chaves) no **JSON** não são identificadores **Pascal** válidos, portanto, o senhor precisa mapear de alguma forma os nomes usados no **JSON** para os nomes de campo e vice-versa.
O campo de data precisa ser lido (e gravado) em um formato que não é tratado por um simples método `StrToDate` ou `DateToStr`. Se vários recursos **REST** forem usados, você precisará de um mapeamento entre a classe e o recurso **REST** a ser consumido.

Uma maneira de fazer isso é criar alguma estrutura de dados que contenha o mapeamento e que forneça o formato. Outra maneira seria fazer com que todas as estruturas de dados descendam de uma classe básica que forneça alguns métodos para criar o mapeamento:

```
TPerson = Class
  class function ResourceName : String;
  class function FieldToJSONName(const aField : String): String;
  class function FormatDate(aDate : TDatetime) : String;
  firstname : string;
  lastname : string;
  birthdate : TDateTime;
end;
```

Se, além disso, você desejar salvar os dados em um banco de dados usando alguma tecnologia **ORM**, isso se tornará ainda mais complicado: Os nomes dos campos do banco de dados também podem ser diferentes dos campos usados na classe. Uma técnica semelhante pode ser usada para introduzir um segundo mapeamento para o banco de dados;

```
TPerson = Class
 Class function TableName : string;
 class function FieldToJSONName(const aField : String): String;
 class function ResourceName : String;
 class function FieldToDBField(const aField : String): String;
 class function FormatDate(aDate : TDatetime) : String;
 firstname : string;
 lastname : string;
 birthdate : TDateTime;
end;
```

O resultado geralmente é uma grande quantidade de funções na declaração (*e sua implementação, é claro*) para cuidar dos mapeamentos necessários e fornecer informações sobre como e o que transmitir.
Os **atributos** podem tornar isso muito mais simples:
As mesmas informações podem ser fornecidas com anotações simples nos próprios campos:

```
[Table('People')]
[Resource('/REST/Contact')]
TPerson = Record
   [DBKeyField]
   id : integer;

   [JSONKey('first-name')]
   [DBField('pe_name')]
   firstname : string;

   [JSONKey('last-name')]
   [DBField('pe_lastname')]
   lastname : string;

   [JSONKey('date-of-birth')]
   [DateFormat('yyyymmdd')]
   [DBField('pe_birthdate')]
   birthdate : TDateTime;
end;
```

O exemplo acima usa 5 atributos:
`Table` especifica a tabela do banco de dados na qual o objeto será salvo.
`Resource` especifica a URL na qual o recurso pode ser recuperado
`JSONKey` especifica a chave a ser usada ao ler/gravar o objeto **JSON**.
`DBField` especifica o nome do campo do banco de dados a ser usado ao carregar/salvar de/para o banco de dados.
`DateFormat` especifica a formatação usada no **JSON** para uma data.

Um sistema que consome serviços **REST** e um segundo sistema que carrega/salva objetos no banco de dados podem inspecionar esses atributos usando RTTI e usá-los para carregar ou salvar os objetos no local correto e com o formato correto. Em um sistema realista, serão necessários mais atributos, é claro.

Se você tiver apenas 1 serviço para criar e uma tabela de banco de dados para manter, o uso de atributos pode ser um exagero: Toda a operação de criação de **JSON** pode, então, ser feita de forma muito mais simples:

```
function TPerson.ToJSON : TJSONObject;
begin
Result:=TJSONObject.Create([
'id',id,
'first-name',firstname,
'last-name',LastName,
'date-of-birth',FormatDateTime('yyyymmdd',birthdate)
]);
end;
```

Mas com muitas tabelas e muitos recursos **REST** para criar ou consumir, o uso de atributos evita escrever muitas funções, como as mencionadas acima, ou até mesmo fornecer os metadados necessários para os sistemas que interagem com o serviço **REST** ou com o banco de dados.
A partir da declaração do objeto, o programador também pode ver imediatamente como os mapeamentos
são definidos, sem a necessidade de mergulhar em funções que fornecem as mesmas informações.
É claro que se pode argumentar que essas metainformações não deveriam estar dentro do próprio objeto,
que esses mapeamentos devem ser construídos fora dos objetos de negócios: a abordagem acima introduz um acoplamento entre a estrutura de streaming de **JSON** usada e os objetos de negócios.
Da mesma forma, para o **ORM** (*Object-Relational Mapping: a estrutura para salvar objetos em um banco de dados*). Além disso, ele não abrange todos os cenários de uso possíveis, por exemplo, e se um objeto precisar ser carregado de um banco de dados e salvo em outro banco de dados com uma estrutura diferente?
Essas perguntas são legítimas, mas são objeto de uma discussão separada:
Aqui, queremos apenas ilustrar um possível uso dos atributos.

❹ A UNIDADE SYSTEM.RTTI

O **RTTI** clássico, tal como existia desde o **Delphi 1** (ou **Free Pascal**), podia ser examinado por meio das rotinas na unidade **`TypInfo`**. Essas rotinas eram de baixo nível e exigiam a alocação manual de memória e o uso de ponteiros. Com a introdução do **RTTI** estendido, também foi introduzida uma nova **API** para consultar e usar o **RTTI**:

A RTTI estendida precisa ser examinada usando a unidade System.Rtti. Ela oferece a mesma funcionalidade que a unidade TypInfo oferecia, mas oferece APIs mais convenientes: registro, objetos.

Para cada tipo e espécie de dados, a unidade RTTI contém um objeto dedicado que representa os dados RTTI para esse tipo. Aqui estão as principais classes:

`TRttiType` Essa é a classe principal de **todos** os tipos.

`TRttiInstanceType` Representa a RTTI para um tipo de **classe**.

`TRttiRecordType` Representa a RTTI de um **registro**.

`TRttiField` Representa a RTTI de um campo de um propriedade ou classe.

`TRttiProperty` Representa a RTTI de uma propriedade de um registro ou classe.

`TRttiMethod` Representa a RTTI de um método de um registro ou classe.

Essas classes oferecem métodos para encontrar RTTI relacionados: para uma classe, sua lista de campos, para um método, seus parâmetros. Quase todas as classes têm um método para obter os **Attributes** associados aos dados ou tipos que representam.

As classes de propriedade e campo têm um método para definir o valor do campo ou da propriedade, dada uma instância. As classes de método podem ser invocadas, permitindo que você chame qualquer método sem conhecer o mecanismo exato de chamada. Para tornar tudo isso possível sem usar os tipos reais, é necessário usar um tipo TValue: esse novo tipo é muito parecido com uma variante, mas oferece uma correspondência de tipo mais exata do que uma variante. Ele é usado para **`set/get valores`** de propriedades ou campos, é usado para retornar valores de funções que você chama e devem ser usados para fornecer valores de parâmetros para métodos que precisam de parâmetros.

As **APIs** desta unidade usam classes, mas o tempo de vida dessas classes é totalmente gerenciado pela unidade: não há necessidade de liberá-las. Para que isso seja possível, um `TRttiContext` deve ser usado ao chamar as APIs da unidade Rtti: quando o contexto for liberado, todas as instâncias das classes RTTI que não são mais usadas serão liberadas.
OBSERVE que, no **Free Pascal**, as rotinas da unidade Rtti são apenas uma camada de conveniência sobre as rotinas da unidade TypInfo: tudo o que pode ser feito com a unidade Rtti também pode ser feito com as rotinas da unidade TypInfo (exceto a funcionalidade Invoke para chamar um método). No Delphi, esse não é o caso: algumas funcionalidades estão presentes apenas na unidade Rtti. Portanto, se você tiver um código que deseja executar no FPC e no Delphi, deve continuar usando a unidade Rtti.

❺ USANDO O RTTI ESTENDIDO E OS ATRIBUTOS

Então, como podemos usar o **Extended RTTI** e os **Attributes**?
Vamos demonstrar isso expandindo o exemplo dado acima:
Criaremos algumas rotinas que criam um objeto JSON a partir dos dados de qualquer registro, após a leitura desse registro no banco de dados. Vimos que foram usados 5 atributos.
Começaremos apresentando o código completo que define o registro da pessoa.

```
unit contact;

  {$mode objfpc}
  {$H+}
  {$modeswitch prefixedattributes}
  {$modeswitch advancedrecords}

interface

uses
  Classes, SysUtils, rest.types, rttitojson.types, objtodb.types;

  {$RTTI EXPLICIT Fields[vcPublic]}
  Type
  [Table('People')]
  [Resource('/REST/Contact')]
  TPerson = Record
    [DBKeyField]
    id : integer;

    [JSONKey('first-name')]
    [DBField('pe_name')]
    firstname : string;

    [JSONKey('last-name')]
    [DBField('pe_lastname')]
    lastname : string;

    [JSONKey('date-of-birth')]
    [DateFormat('yyyymmdd')]
    [DBField('pe_birthdate')]
    birthdate : TDateTime;
  end;
```

Um atributo extra (`DBKeyField`) é usado para indicar o campo-chave do registro.

O bloco com as diretivas modeswitch instrui o compilador **Free Pascal** a permitir atributos prefixados - Isso é necessário porque, tradicionalmente, o **Free Pascal** usa a notação de atributo para especificar modificadores para procedimentos e funções. A segunda diretiva permite o uso de registros avançados. A diretiva *{$RTTI }* merece uma menção: Essa diretiva controla a quantidade de RTTI gerada para suas classes e registros.

{$RTTI EXPLICIT FIELDS[vcPublic]}

A keyword **`EXPLICIT`** informa ao compilador que o que vem a seguir é o único tipo de informação que deve ser gerado e que ele deve ignorar as configurações de **RTTI** das classes pai.

Como alternativa, INHERITED pode ser usada para indicar que o que se segue deve ser gerado além das informações usadas para a classe pai. Por fim, a palavra-chave FIELDS informa ao compilador que tipo de informação deve ser gerada para os campos. As outras possibilidades são METHODS e PROPERTIES, que são usadas para especificar qual RTTI deve ser gerada para métodos e propriedades, respectivamente. Por fim, uma lista de visibilidades deve ser especificada: as seções de visibilidade para as quais as informações de RTTI devem ser geradas.
A diretiva acima, portanto, informa ao compilador que ele só deve gerar RTTI para campos públicos.
A diretriz a seguir diz ao compilador para gerar todas as informações possíveis de RTTI

```
{$RTTI EXPLICIT
FIELDS      [ vcPrivate, vcProtected, vcPublic, vcPublished]
METHODS     [ vcPrivate, vcProtected, vcPublic, vcPublished]
PROPERTIES[ vcPrivate, vcProtected, vcPublic, vcPublished]
}
```

O próximo aspecto a ser observado é a presença das unidades rest.types, rtitojson.types e objtodb.types. Essas unidades definem os vários atributos usados na definição.
Um atributo no Delphi é qualquer classe que descende de TCustomAttribute.
A notação de atributo

```
[JSONKey('first-name')].
```

informa ao compilador que ele deve construir um atributo da classe `JSONKey` ou `JSONKeyAttribute`, (*o compilador opcionalmente removerá o atributo do nome da classe ao procurar a classe*) e que ele deve passar a cadeia de caracteres 'first-name' para o construtor da classe.
A unidade a seguir define os atributos para o **JSON streaming:**:

```
unit rttitojson.types;
{$mode ObjFPC}{$H+}
interface

uses
  Classes, SysUtils;
Type
{ JSONKeyAttribute }
JSONKeyAttribute = class(TCustomAttribute)
private
  FKey: string;
public
  constructor create(aKey : String);
  property Key : string Read FKey;
end;
{ DateFormatAttribute }
DateFormatAttribute = class(TCustomAttribute)
  FFormat : String;
Public
  constructor create(aFormat : String);
  Property DateFormat : String Read FFormat;
end;
```

A implementação é bastante simples:

```
implementation

{ JSONKeyAttribute }
constructor JSONKeyAttribute.create(aKey: String);
begin
  FKey:=aKey;
end;

{ DateFormatAttribute }
constructor DateFormatAttribute.create(aFormat: String);
begin
  FFormat:=aFormat;
end;

end.
```

A chamada `GetAttributes` pode ser usada para obter todos os atributos de um identificador. A chamada `GetAttributes` pode ser usada para obter um único atributo, especificando a `class` do `attribute`.

❻ USANDO ATRIBUTOS PARA CRIAR JSON

Como podemos usar isso para converter um registro em um objeto **JSON**? Isso é feito com um `TRTTIJSONWriter` em uma unidade separada (*obviamente, também poderíamos ter usado uma classe*). Observe que o registro `TPerson` não faz referência ao escritor: somente os atributos são necessários para definir nosso `record TPerson`.

```
unit rttitojson.writer;
{$mode ObjFPC}
{$H+}
{$modeswitch advancedrecords}
interface

uses
  Classes, SysUtils, fpJSON, typinfo, rtti, rttitojson.types;

Type
  TRTTIJSONWriter = record
private
  Ctx : TRTTIContext;
  function DateFieldToJSON(aDate: TDateTime; Fld: TRTTIField): TJSONData;
  function FieldToJSON(aData: Pointer; Fld: TRTTIField): TJSONData;
Public
  Procedure ToJSON(aData : Pointer; aTypeInfo : PTypeInfo; aJSON : TJSONObject);
  Generic function ToJSON<T>(var aData : T) : TJSONObject;
  class function create : TRTTIJSONWriter; static;
  procedure Free;
end;
```

Os métodos create e free criam e liberam um `TRttiContext` que é necessário para chamar as várias functions da unidade **Rtti**.

```
class function TRTTIJSONWriter.create: TRTTIJSONWriter;
begin
  Result.Ctx:=TRttiContext.Create(False);
end;

procedure TRTTIJSONWriter.Free;
begin
  Ctx.Free;
end;
```

O ponto de entrada principal é a função genérica ToJSON. Em notação Free Pascal:

```
Generic function ToJSON<T>(var aData : T) : TJSONObject;
```

Na verdade, essa é apenas uma função de conveniência com uma implementação bastante simples. Ela cria o objeto **JSON** resultante e chama uma versão sobrecarregada da function `ToJSON`, passando-lhe as informações de tipo do **type T** (um record):

```
generic function TRTTIJSONWriter.ToJSON<T>(var aData : T) : TJSONObject;
begin
  Result:=TJSONObject.Create;
  try
    ToJSON(@aData,TypeInfo(T),Result);
  except
    Result.Free;
    Raise;
  end;
end;
```

O trabalho real é feito na outra `function ToJSON`.
Basicamente, ela obtém o **Rtti** do registro e faz um loop por todos os campos:
a lista de campos é recuperada com a chamada `GetFields`, que retorna uma matriz de instâncias de `TRttiField`. Para cada campo, ele determina o nome da chave a ser usado ao gravar o **JSON** e, em seguida, chama `FieldToJSON` para converter o valor do campo em um elemento de dados **JSON**, que é então adicionado ao `object JSON`.

```
procedure TRTTIJSONWriter.ToJSON(aData: Pointer; aTypeInfo: PTypeInfo;
                                                 aJSON: TJSONObject);
Var
  R   : TRttiRecordType;
  Fld : TRttiField;
  Key : JSONKeyAttribute;
  KeyName : String;
begin
  R:=Ctx.GetType(aTypeInfo) as TRttiRecordType;
  For Fld in R.GetFields do
    begin
      KeyName:=Fld.Name;
      Key:=JSONKeyAttribute(Fld.GetAttribute(JSONKeyAttribute));
      if Assigned(Key) then
        KeyName:=Key.Key;
      aJSON.Add(KeyName,FieldToJSON(aData,Fld));
    end;
end;
```

OBSERVE o uso da chamada **`GetAttribute`** com o nome de classe **`JSONKeyAttribute`**. Se não houver nenhum atributo dessa classe existir para o campo, Nil será retornado e, nesse caso, o nome real do campo será usado. A chamada **`FieldToJSON`** começa obtendo o valor do campo usando o nome **`GetValue`**, passando à função um ponteiro para o registro. A **`função GetValue`** usará o **RTTI** do campo para calcular o endereço de memória correto para recuperar o valor:

```
function TRTTIJSONWriter.FieldToJSON(aData: Pointer; Fld : TRTTIField): TJSONData;
var
  V : TValue;
begin
  V:=Fld.GetValue(aData);
  case V.Kind of
    tkFloat : if V.TypeInfo=TypeInfo(TDateTime) then
                Result:=DateFieldToJSON(V.AsDateTime,Fld)
              else
                Result:=TJSONFloatNumber.Create(V.AsDouble);
    tkInt64 : Result:=TJSONInt64Number.Create(V.AsInt64);
    tkInteger : Result:=TJSONIntegerNumber.Create(V.AsInteger);
    tkEnumeration : Result:=TJSONString.Create(GetEnumName(V.TypeInfo,V.AsInteger));
    tkAString,
    tkString : Result:=TJSONString.Create(V.AsString);
    tkWString,
    tkUString : Result:=TJSONString.Create(UTF8Encode(V.AsUnicodeString));
  else
    Raise Exception.Create('Unsupported type');
  end;
end;
```

Depois que o valor for recuperado, o tipo do valor será examinado e uma estrutura de dados **JSON** apropriada será criada usando os dados.
Para um campo `TDateTime`, uma rotina especial é chamada que usa o atributo `DateFormat` para criar uma string **JSON** corretamente formada:

```
function TRTTIJSONWriter.DateFieldToJSON(aDate : TDateTime; Fld : TRTTIField) : TJSONData;
var
  Res,Fmt : string;
  DateFormat : DateFormatAttribute;
begin
  Fmt:='';
  DateFormat:=DateFormatAttribute(Fld.GetAttribute(DateFormatAttribute));
  if Assigned(DateFormat) then
    Fmt:=DateFormat.DateFormat;
  if Fmt='' then
    Res:=DateToISO8601(aDate)
  else
    Res:=FormatDateTime(Fmt,aDate);
  Result:=TJSONString.Create(Res);
end;
```

Com isso, estamos prontos. O programa a seguir testa nosso código:

```
uses sysutils, contact, fpjson, rttitojson.types, rttitojson.writer;

Procedure WriteJSON(P : TPerson);
var
  Writer  : TRTTIJSONWriter;
  JSON    : TJSONObject;
begin
  JSON    := Nil;
  Writer  := TRTTIJSONWriter.Create;
  try
    JSON   :=Writer.ToJSON<TPerson>(P);
    Writeln('JSON : ',JSON.FormatJSON());
  finally
    JSON.Free;
    Writer.Free;
  end;
end;

var Person : TPerson;

begin
  With Person do
    begin
      id:=1;
      FirstName:='Kirth';
      LastName:='Gersen';
      birthdate:=EncodeDate(1486,5,14);
    end;
  WriteJSON(Person);
end.
```

O resultado é mostrado na figura 1 na página 9 deste artigo.

```
File  Edit  View  Search  Terminal  Tabs  Help
(michael) home: /home/michael/source/articles/extrtti
home: ~/source/articles/extrtti
> ./demo
JSON : {
  "id" : 1,
  "first-name" : "Kirth",
  "last-name" : "Gersen",
  "date-of-birth" : "14860514"
}
home: ~/source/articles/extrtti
>
```

Figura 1: O JSON gerado

❼ USANDO ATRIBUTOS PARA LER E GRAVAR NO BANCO DE DADOS

Uma unidade semelhante com um registro semelhante pode ser criada para a `reading/writing` de um objeto do banco de dados:

```
TRTTIDBReader = record
private
  Ctx : TRTTIContext;
  Conn : TSQLConnection;
  function CreateQuery(aSQL : String) : TSQLQuery;
  Procedure DBFieldToField(DBData : TDataset; aData: Pointer; Fld: TRTTIField);
  Procedure FieldsToParams(Params : TParams; aData: Pointer; Rec : RttiRecordType);
  function RecordToWhereClause(Rec: TRttiRecordType): String;
  Function RecordToSQL(Rec: TRttiRecordType) : String;
  procedure ValueToParam(V: TValue; Parm: TParam);
Public
  function ReadFromDB(aData : Pointer; aTypeInfo: PTypeInfo) : Boolean;
  Generic function ReadFromDB<T>(var aData : T) : Boolean;
  class function create(aConnection : TSQLConnection): TRTTIDBReader; static;
  procedure Free;
end;
```

Examinaremos apenas os métodos principais. A função genérica é novamente um pequeno invólucro de conveniência:

```
Generic function TRTTIDBReader.ReadFromDB<T>(var aData : T) : Boolean;
begin
   Result:=ReadFromDB(@aData,TypeInfo(T));
end;
```

O trabalho real é feito na função a seguir. Ela começa usando a rtti do registro para criar uma instrução **`SQL SELECT`**, que é então usada para criar um objeto de consulta: a consulta é aberta e, se ela retornar um resultado, o registro é carregado a partir do resultado:

```
Function TRTTIDBReader.ReadFromDB(aData : Pointer; aTypeInfo: PTypeInfo) : Boolean;
var
  lRtti : TRttiRecordType;
  Qry   : TSQLQuery;
  Fld   : TRttiField;
  SQL   : String;
begin
  lRtti := Ctx.GetType(aTypeInfo) as TRttiRecordType;
  SQL   := RecordToSQL(lRtti);
  Qry   := CreateQuery(SQL);
  try
    FieldsToParams(Qry.Params,aData,lRtti);
    Qry.Open;
    Result := not Qry.IsEmpty;
    if Result then
      For Fld in lRtti.GetFields do
          DBFieldToField(Qry,aData,Fld);
    finally
    Qry.Free;
  end;
end;
```

O loop principal aqui é novamente um loop sobre os campos do registro. O procedimento `DBFieldToField` transfere os dados dos campos do conjunto de dados para os campos do registro.

A rotina `RecordToSQL` constrói uma instrução **`SQL Select`** para recuperar os dados para o registro do banco de dados. Ela usa o atributo `Table` na definição do registro para determinar o nome da tabela e usa o atributo `DBField` nos campos da definição do registro para obter o nome dos campos da tabela do banco de dados. Em ambos os casos, o nome pascal do tipo de registro ou campo tipo de registro ou campo é usado como um fallback:

```
Function TRTTIDBReader.RecordToSQL(Rec: TRttiRecordType) : String;
var
  Tbl : TableAttribute;
  Fld : TRttiField;
  DBField : DBFieldAttribute;
  FieldName : String;
  TableName : string;

begin
  Result:='';
  For Fld in Rec.GetFields do
    begin
      FieldName:=Fld.Name;
      DBField:=DBFieldAttribute(Fld.GetAttribute(DBFieldAttribute));
      if Assigned(DBField) then
        FieldName:=DBField.Name;
      if Result<>'' then
        Result:=Result+', ';
      Result:=Result+FieldName;
    end;
  TableName:=Rec.Name;
  Tbl:=TableAttribute(Rec.GetAttribute(TableAttribute));
  If Assigned(Tbl) then
    TableName:=Tbl.Table;
  Result:='SELECT '+Result+' FROM '+TableName;
  Result:=Result+' WHERE '+RecordToWhereClause(Rec);
end;
```

A `function RecordToWhereClause` executa um loop semelhante para construir a cláusula Where que selecionará um único registro do banco de dados, com base nos key fields, que são marcados com o atributo `DBKeyField`:

```
on TRTTIDBReader.RecordToWhereClause(Rec: TRttiRecordType) : String;
var
  Fld : TRttiField;
  DBField : DBFieldAttribute;
  FieldName : string;
begin
  Result:='';
  For Fld in Rec.GetFields do
  begin
    FieldName:=Fld.Name;
    if Assigned(Fld.GetAttribute(DBKeyFieldAttribute)) then
    begin
      DBField:=DBFieldAttribute(Fld.GetAttribute(DBFieldAttribute));
      if Assigned(DBField) then
        FieldName:=DBField.Name;
      if Result<>'' then
        Result:=Result+' AND ';
      Result:=Result+'('+ FieldName+' = :'+FieldName+')';
    end;
  end;
end;
```

A definição de registro apresentada anteriormente lê os registros da tabela `People`, que pode ser definida em **SQL** da seguinte forma:

```
create table people (
id int not null,
pe_name varchar(127) not null,
pe_lastname varchar(127) not null,
pe_birthdate date not null
);
```

As rotinas acima criarão o seguinte **SQL**:

```
SELECT id, pe_name, pe_lastname, pe_birthdate
FROM people WHERE (id = :id)
```

Para preencher os parâmetros usados na cláusula **WHERE**, é usada a seguinte rotina, que é apenas outra variação das rotinas anteriores:

```
Procedure TRTTIDBReader.FieldsToParams(Params: TParams; aData: Pointer;
                                       Rec: TRTTIRecordType);
var
  Fld       : TRttiField;
  DBField   : DBFieldAttribute;
  Parm      : TParam;
  V         : TValue;
  FieldName : string;

begin
  For Fld in Rec.GetFields do
    begin
      FieldName:=Fld.Name;
       if Assigned(Fld.GetAttribute(DBKeyFieldAttribute)) then
         begin
           DBField:=DBFieldAttribute(Fld.GetAttribute(DBFieldAttribute));
           if Assigned(DBField) then
             FieldName:=DBField.Name;
           Parm:=Params.FindParam(FieldName);
           if Assigned(Parm) then
           begin
             V:=Fld.GetValue(aData);
             ValueToParam(V,Parm);
           end;
         end;
    end;
end;
```

A rotina `ValueToParam` aplica o `TValue` a um `TParam`. Nenhum atributo é usado
Desta vez, o `TValue` contém todas as informações de que precisamos para aplicar o valor a um parâmetro de consulta

```
Procedure TRTTIDBReader.ValueToParam(V : TValue; Parm : TParam);
begin
  Case V.Kind of
    tkInteger : Parm.AsInteger:=V.AsInteger;
    tkInt64 : Parm.AsInteger:=V.AsInt64;
    tkAString,
    tkString : Parm.AsString:=V.AsString;
    tkUString : Parm.AsUnicodeString:=V.AsUnicodeString;
    tkWString : Parm.AsUnicodeString:=V.AsUnicodeString;
    tkFloat :
      if V.TypeInfo=TypeInfo(TDateTime) then
        Parm.AsDateTime:=V.AsDateTime
      else
        Parm.AsFloat:=V.AsDouble;
  else
    Raise Exception.Create('Unsupported type')
  end;
end;
```

Por fim, a rotina DBFieldToField é usada para gravar o conteúdo de um campo do banco de dados em um campo do registro.
O método TRttiField.SetValue aceita um valor de TV, portanto a rotina começa criando esse valor a partir do campo do banco de dados:

```
Procedure TRTTIDBReader.DBFieldToField(DBData : TDataset; aData: Pointer;
          Fld: TRTTIField);
var
   V : TValue;
   DBField : DBFieldAttribute;
   DBFld : TField;
   FieldName, S : String;
   DT : TDateTime;
   U : UnicodeString;
begin
   FieldName:=Fld.Name;
   DBField:=DBFieldAttribute(Fld.GetAttribute(DBFieldAttribute));
   if Assigned(DBField) then
      FieldName:=DBField.Name;
   DBFld:=DBData.FieldByName(FieldName);
   Case DBFld.DataType of
      ftInteger,
      ftSmallint,
      ftWord : TValue.Make(DBFld.AsInteger,fld.FieldType.Handle,V);
      ftLargeInt : TValue.Make(DBFld.AsLargeInt,fld.FieldType.Handle,V);
      ftString :
      begin
         S:=DBFld.AsString;
         TValue.Make(@S,fld.FieldType.Handle,V);
      end;
      ftWideString:
      begin
         U:=DBFld.AsUnicodeString;
         TValue.Make(@U,fld.FieldType.Handle,V);
      end;
      ftDate
      ftTime,
      ftDateTime:
        begin
          DT:=DBFld.AsDateTime;
          TValue.Make(@DT,fld.FieldType.Handle,V);
        end;
   else
    Raise Exception.Create('Unknown field type: '+GetEnumName(TypeInfo(TFieldType);
 end;
   Fld.SetValue(aData,V);
end;
```

Depois que o valor é criado, o `Fld.SetValue` copia o valor para o campo apropriado do registro.
Com tudo isso em mãos, o código para ler um registro de um banco de dados e gravá-lo como uma estrutura estrutura **JSON** é o seguinte: (*veja a próxima página*)

```
Function LoadPerson(aID : Integer) : TPerson;
var Reader : TRTTIDBReader;
    Conn   : TSQLConnection;
    P      : TPerson;
    ReadOK : Boolean;
begin
  P.ID:=aID;
  conn:=GetConnection;
  try
    Reader:=TRTTIDBReader.Create(Conn);
    ReadOK:=Reader.ReadFromDB<TPerson>(P);
    if not ReadOK then
     Writeln('Failed to read person');
     Result:=P;
  finally
    Conn.Free;
  end;
end;

begin
  WriteJSON(LoadPerson(1));
end;
```

O mesmo código poderia ser usado para carregar e gravar todos os outros objetos. Se inserirmos um registro no banco de dados com o seguinte **SQL**:

```
insert into people values (1,'Kirth','Gersen','1486-05-14');
```

Então, o resultado do nosso programa modificado será exatamente igual ao da figura 1 (*na página 9 deste artigo*).

❽ CONCLUSÃO

A chegada do **Extended RTTI** no Free Pascal abre novas possibilidades no **Free Pascal** e no **Lazarus:** Como mostrado aqui, **é possível criar uma serialização JSON funcional** e uma miniestrutura ORM com muito pouco código. Não é preciso dizer que, em um programa do mundo real, serão necessários mais alguns atributos, mas o funcionamento básico seria o descrito aqui. Nem por isso a chegada do **Extended RTTI** no **Free Pascal** permitirá o uso no Free Pascal de estruturas semelhantes escritas para o **Delphi**.

A CHEGADA DO EXTENDED RTTI NO FREE PASCAL PERMITE O USO DE FRAMEWORKS ESCRITOS PARA O DELPHI.

JUN 13-14 2024 | AMSTERDAM

Spinnekop 3, 1444 GN Purmerend, the Netherlands

Delphi Summit

QUANDO TRABALHÁVAMOS EM CASA

POR IAN BARKER, DEVELOPER ADVOCATE DA EMBARCADERO.

A pandemia global de 2020 foi algo que causou muitos danos. Vidas perdidas, meios de subsistência perdidos, de fato, muitas perdas. Com certeza, isso será escrito nos futuros livros de história. A vida mudou também em uma infinidade de outras formas. Muitas empresas adotaram o conceito de cenários remotos e de trabalho em casa. Algumas por ganância, ou necessidade, de continuar ganhando dinheiro, enquanto o mundo aprendeu que mãos tossindo que tocam frutas tocam o senhor, tocam seus funcionários e, por fim, tocam suas finanças. A maioria, é claro, simplesmente seguiu os conselhos científicos de ficar longe uns dos outros, evitar espaços públicos lotados e desprezar o deslocamento diário.

À medida que a COVID se desvanece na história, há um pouco de reversão acontecendo, especialmente no setor de tecnologia, com "ajustes no número de funcionários" em massa. Muitos gigantes da tecnologia que se gabavam de permitir que a equipe trabalhasse em casa, permanentemente ou, pelo menos, principalmente, renegaram essa ideia e vacilaram no altar de uma crença corporativa fundamental de que, para ser eficiente, é melhor agrupar o maior número possível de pessoas em um único espaço físico - o escritório.
Quanto maior a empresa, maior a probabilidade de ter um prédio enorme, semelhante a uma baleia, lotado de pessoas que poderiam ter recuperado algumas horas de seu tempo pessoal se não estivessem lá e, em vez disso, participassem de reuniões virtualmente. Parece que algumas lições aprendidas durante a pandemia foram esquecidas muito rapidamente.

ACIONISTAS DIZEM QUE O PRESENCIAL É MELHOR

Se o senhor for um desenvolvedor de software, garanto que não será mais eficiente se tiver que se deslocar para um escritório e passar horas enfiado em um cubículo, na eventualidade de uma reunião em que sua presença física possa, de alguma forma, melhorá-la. Digo isso como desenvolvedor sênior com quase 40 anos de experiência em todos os níveis, em todos os tipos de empresas, desde pequenas empresas iniciantes até gigantes da Fortune 500. É engraçado como a decisão "no escritório, não em casa" é muitas vezes tomada por líderes que não entram no prédio onde os grandes não lavados exercem seu ofício ou, na verdade, têm o que qualquer pessoa reconheceria como um trabalho regular de escritório.

Com o advento de softwares de reunião capazes, como o Google Meet, Skype, Microsoft Teams, Zoom, GoToMeeting e muitos outros concorrentes, combinados com webcams e microfones ou fones de ouvido de alta qualidade e a preços acessíveis, não há quase nada que exija que o senhor esteja fisicamente presente se for desenvolvedor de software, designer, gerente de projeto ou controle de qualidade. O senhor não é um padeiro, não precisa cheirar fisicamente seus produtos. O senhor não é um vinicultor, portanto, não há necessidade de subir e descer hectares de vinhedos, resmungando em voz alta sobre a umidade do solo e a friabilidade do seu terroir. Seu trabalho existe literalmente no éter. O senhor É a Matrix. Esse é o lugar certo para o senhor.

Há muitas opções também se for uma questão de confiança. Há muitos aplicativos que avaliam a atividade do seu funcionário, verificam se ele não está se divertindo ou "Netflix e relaxando" quando deveria estar consertando bugs. Se o senhor precisa ser um chefe nerd e maníaco por controle, há todo um setor de aplicativos de software de boatos que competem para ser seu espião número um.

ÀS VEZES, O SENHOR SÓ PRECISA DE BRINDES E DE SAIR

Mas algumas coisas são melhores se feitas pessoalmente e uma delas foi dizimada pela pandemia: a conferência de tecnologia presencial. O senhor pode participar de qualquer tipo de evento virtual que quiser, mas posso lhe dizer com certeza que as conexões casuais que o senhor faz com os técnicos que pensam da mesma forma são muito mais profundas e duradouras se forem presenciais. É verdade que, como espécie, nós, geeks e nerds, temos uma certa reputação de falta de higiene pessoal e um senso de vestimenta que poderia cegar um unicórnio a 100 metros de distância, mas acho que isso é algo imerecido... na maioria das vezes.

Quando o mundo finalmente se acostumou com a COVID-19, todos nós voltamos a nos preocupar com coisas normais, como uma guerra nuclear que nos eliminaria, em vez de nos preocuparmos com as maçanetas sujas da porta de um supermercado. Pouco a pouco, os eventos presenciais voltaram. Em outubro passado, pude participar do evento Foren Tage na Alemanha. Desde então, tem havido um aumento constante nas reuniões e conferências presenciais.

O GLOBAL DELPHI SUMMIT 2024

Se os senhores puderem ir a Amsterdã nos dias 13 e 14 de junho, recomendo que participem do Global Delphi Summit, organizado pela GDK Software e por vários patrocinadores, incluindo o sempre maravilhoso Barnsten e, é claro, a Embarcadero.

Eu estarei lá nos dois dias, o dia todo. Apresentarei palestras no primeiro e no segundo dia, mas também estarei por lá, encontrando todos os senhores e respondendo às suas perguntas sempre que possível. O imperdível Marco Cantú apresentará uma sessão especial e outras pessoas conhecidas do Delphi, como o Dr. Bob, também estarão presentes. Vai ser realmente incrível.

Sei que nem todos poderão ir. Sei como o senhor se sente, pois também já estive nessa situação muitas vezes e é doloroso. Farei streaming ao vivo durante os dois dias, e haverá também um streaming oficial e gravações feitas pelos organizadores. Se o senhor estiver participando em junho, venha dizer oi.

Delphi Summit
JUN 13-14 2024 | AMSTERDAM

Delphi Summit

JUN 13-14 2024 | AMSTERDAM

embarcadero®
Official Technology Partner

For just 249,-* you will get:

- Admission for both days, with all speakers
- Each day from 10:00 a.m. to 6:00 p.m.
- Lunch and drinks included
- Free video recording of all sessions
- All sessions in English
- Free parking + easy public transport from Amsterdam Schiphol Airport
- Hotel rooms can be booked separately after ordering
- Discount with two or more tickets

***Early bird discount, ends April 1st**

https://delphisummit.com

Extra 10% discount for Blaise Pascal readers! Use code AVB16BT6S1BF at the check-out

See you there!

Delphi Summit
JUN 13-14 2024 | AMSTERDAM

embarcadero
Official Technology Partner

Delphi Summit 2024

30+ **In Depth Sessions**
all about Delphi and Pascal

300 **Attendees**
meeting fellow developers

2 **Full Delphi Days**
from 10 PM to 6 PM

15+ **Acclaimed speakers**
from all over the world

THE DELPHI SUMMIT IN A NUTSHELL

About the summit.

The summit is organised by GDK Software, in partnership with Embarcadero and Barnsten. It is a two-day event packed with the latest and greatest news and innovations, all about our favourite programming language: Delphi.

LOCATION AND SPEAKERS

Meet and great

The location is the H20 Esports conference centre, located near Amsterdam, The Netherlands. Speakers include Jim McKeeth, Ian Barker, Marco Cantu and many more!

Join the Delphi Summit: https://delphisummit.com

Delphi Summit 2024

Agenda

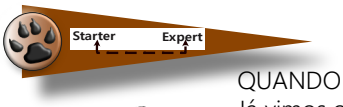
Amsterdam. June 13-14, 2024

Day 1: Thursday 13th

09:00	**Registration**		
	Main stage		
10:00	**Welcome and opening – Kees de Kraker and Marco Geuze**		
10:15	**Keynote – Jim McKeeth and Ian Barker**		
11:00	Coffee Break & Network		
	Stage Sydney	Stage Alexandria	Stage Rio
11:30	**Cary Jensen** - Selected Advanced FireDAC Technologies FireDAC supports a wide range of powerful and useful operations. This session will discuss and demonstrate four of the more interesting ones, including caching updates, batch move operations, using FireDAC built-in functions, and Local SQL.	**Fabrizio Bitti -** Creating a real-life Blockchain with Delphi Demonstrate how a blockchain works and what it is used for. All with Delphi in a multithread environment to mine the blocks.	**Steffen Nyeland -** I can, therefore IAM Changing your application login process to an IAM (Identity and Access Management) controlled process
12:30	Lunch, Network & Gaming		
13:30	**Marco Cantu** - Building FireMonkey Apps with Style Unlike VCL, styles in FireMonkey don't only determine the graphical elements of a UI control, but also its architecture. In this session, we'll explore how styles work, how to customize controls at runtime, how to build new styled FMXcomponents, and how this all helps building a single-source multi-device UI.	**Richard Hatherall –** AWS SDK	**Serge Pilko** – How to replace DataBase components with Rest API calls in Delphi An introduction to REST and creating a cross-platform REST Client application, using Embarcadero's REST Client library to replace database access components.
14:30	**Jim McKeeth -** Evidence Based Delphi Engineering Why do you write code *that way*? Chances are it is “the way you’ve always done it.” Learn how to gather the evidence you need to know the *right way*.	**Frank Lauter** - MVVM - The Delphi Way! A waste of time or a way to keep the source code maintainable? Frank Lauter will present his view on the MVVM pattern and explain which steps are necessary for new and legacy applications.	**Primož Gabrijelčič** - Defensive programming Learn from someone with 35 years of experience how to write code that will be easy to understand now, and in the future. Dive into some of my own ,laughable, terrible code examples with me and get easy-to-reuse advice on how to improve
15:30	Break & Network		
16:00	**Marco Geuze** – Delphi and AI Large language models (LLMs) provide significant help for development. Learn how to use a private LLM in Delphi without giving away your privacy and source code.	**Bob Swart -** REST with WebBroker in Delphi	**Conrad Vermeulen** - From monoliths to microservices Building composable applications with a bit of runtime config
17:00	Network & Gaming		
18:00	Day 1 ends		

Day 2: Friday 14th

09:00	**Registration**		
	Mainstage		
10:00	**Welcome and opening – Kees de Kraker and Marco Geuze**		
10:15	**Panel discussion with Jim McKeeth, Marco Cantu, Ian Barker and MVPs**		
11:00	Coffee Break & Network		
	Stage Sydney	**Stage Alexandria**	**Stage Rio**
11:30	**Kees de Kraker** – Test-driven development Using the TTD approach in a monolith application is not feasible, right? Join this session to discover the impossible.	**Patrick Quist** – Linux Delphi Services A journey through the Cloud(s)	**Stefan Glienke** – Spring4D Some goodies from the Spring4D collections.
12:30	Lunch, Network & Gaming		
13:30	**Olaf Monien -** REST Easy Connecting to REST APIs and visualizing data on desktop and mobile devices.	**Christoph Schneider** – Firestone Cloud For the Firestore Cloud database, the FB4D open-source library contains everything you need to access it from VCL/FMX applications. In this session, the author will show you how easy it is to write and read a document and to be notified of changes in the database with the new object-to-document wrapper.	**Patrick Prémartin** – Our users want to access their data from anywhere, on any type of device, with or without an Internet connection. Some also want to work together offline or online, remotely or on-site, on desktops, laptops, smartphones or tablets. Here's an easy-to-implement solution in Delphi to transform any local database into a synchronized one
14:30	**Carlos Agnes** - The Best of Delphi Underground A set of small Delphi secrets and how they work under the hood. IDE and debugging tips, historical issues like why the base date for TDateTime is 12/30/1899, Exceptions stacks, interface tricks, and the dictionary of secrets.	**Andrea Raimondi** - Algorithmic password hardening From the forgotten lessons of Enigma to generating salts and scrambling passwords, Andrea will guide you through the best ways to keep everything safe.	**Ray Konopka** - Component Building: Fundamentals This session focuses on the fundamental techniques required for building robust Delphi components. We build a custom component, showing the key classes from which all components descend, followed by an analysis of the anatomy of a component. We conclude with a discussion on the proper way to distribute custom components through runtime and design packages.
15:30	Break & Network		
15:45	To be announced		
	Mainstage		
16:15	**Ian Barker -** What to do if you're old, ugly, and everything is annoying Join Ian for this session where he applies his uniquely lively style of presentation to the subject of software development in an age where everyone wants your apps to be free, have a name like ZZxQFlmbl, and be 'monetized' by a YouTube influencer with green hair, a pierced fingernail, and their own brand of hair removal creme.		
16:45	**Door prize giveaway**		
17:00	**Closing talk with Jim McKeeth, Kees de Kraker and Marco Geuze**		
17:15	Network & Gaming		
18:00	Conference ends		



QUANDO FOR A HORA CERTA

Já vimos os pontos de interrupção antes. Eles nos dão a capacidade de executar nosso aplicativo e interromper qualquer código suspeito que queiramos inspecionar com o debugger.

Mas e se esse código estiver em um loop ou for chamado dentro de um loop?
O ponto de interrupção será atingido em cada iteração do loop e o erro poderá ocorrer somente após muitas iterações. Isso pode ocorrer centenas de vezes e, por isso, temos de pressionar **F9** para continuar. Há maneiras melhores de lidar com isso. Podemos dizer ao debugger quando ele deve e quando não deve pausar o aplicativo em um ponto de interrupção.

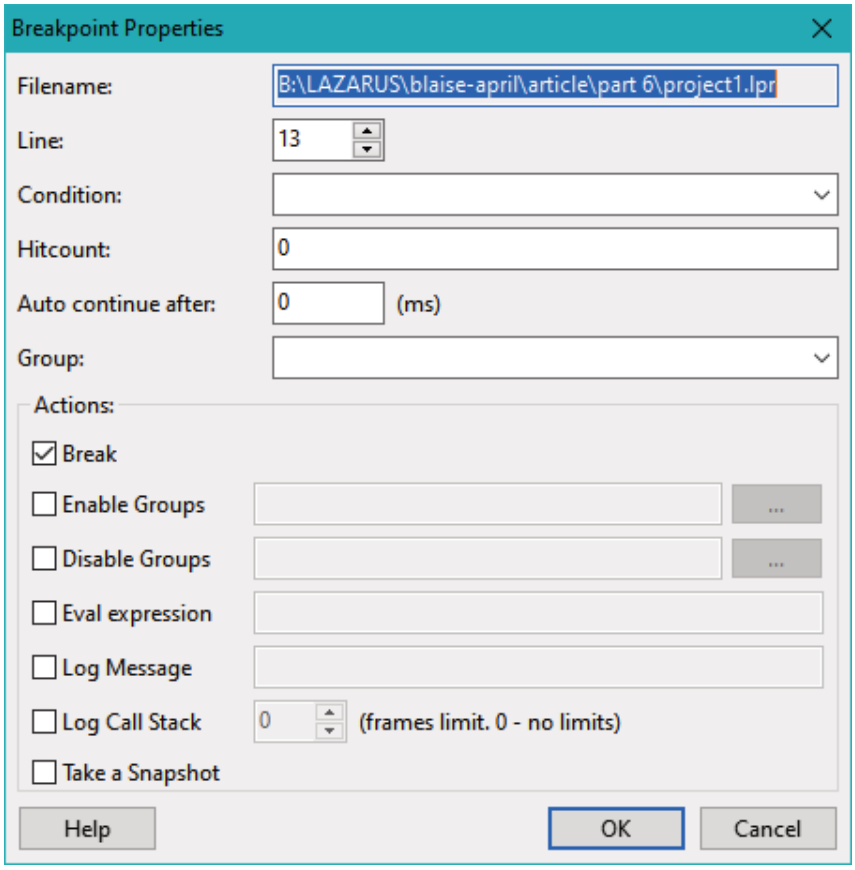
O aplicativo a seguir converte números hexadecimais em inteiros, e nós o testamos chamando o método com todos os números hexadecimais de **$000** a **$FFF**. O resultado deve ser o mesmo que o da compilação do FPC em "`StrToInt`".

```
1. program project1;
2. {$Mode objfpc}{$H+}
3.
4. uses SysUtils;
5.
6. function Hex2Int(h: string): integer;
7. const
8.   c = '0123456789ABCDEF';
9.   d: array [0..3] of integer = (1, 16, 265, 4096);
10. var
11.   i, j: Integer;
12. begin
13.   Result := 0;
14.   for i := 0 to Length(h)-1 do begin
15.     j := pos(h[Length(h)-i], c) - 1;
16.     Result := Result + j * d[i];
17.   end;
18. end;
19.
20. procedure Test;
21. var
22.   t,t1,t2,t3: String;
23.   i: Integer;
24. begin
25.   for t1 in ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'] do
26.   for t2 in ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'] do
27.   for t3 in ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'] do
28.   begin
29.     t := t1+t2+t3;
30.     i := Hex2Int(t);
31.     WriteLn('$'+t, ' = ', i);
32.     if i <> StrToInt('$'+t) then
33.       exit;
34.   end;
35. end;
36.
37. begin
38.   Test;
39.   readln;
40. end.
```

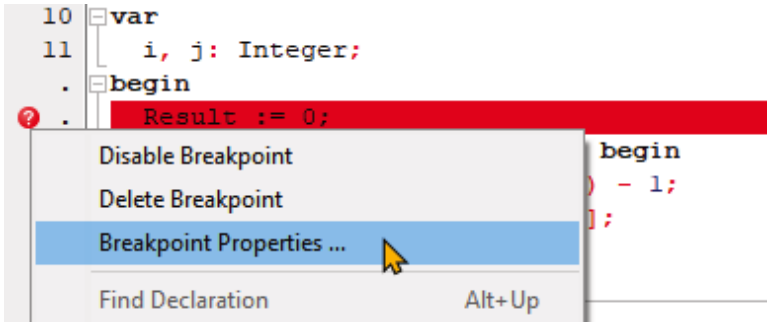
O aplicativo imprimirá muitas linhas e depois parará com

```
$0FE = 254
$0FF = 255
$100 = 265
```

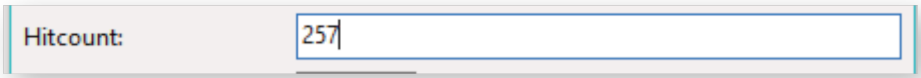

Deveria ter sido executado até $FFF, mas nosso Hex2Int retornou o valor errado para **$100**.
A simples definição de um ponto de interrupção no Hex2Int nos obrigará a continuar **256** vezes antes de chegarmos a esse valor. Portanto, precisamos de outra abordagem. É claro que o aplicativo de exemplo é muito simples, e poderíamos simplesmente iniciar o loop no valor de que precisamos, mas em um aplicativo da vida real, talvez não tenhamos essa opção. Então, vamos fingir que temos que executar todas as iterações antes de podermos utilizar o debuger no problema.
A seguir, três exemplos que mostram maneiras diferentes de parar apenas na iteração com erros. Cada um deles usa uma propriedade diferente do ponto de interrupção. Todas as propriedades podem ser definidas na caixa de diálogo a seguir:



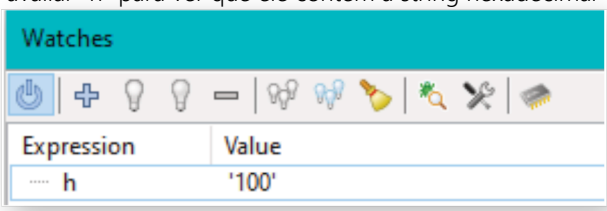
A própria caixa de diálogo pode ser aberta no menu de contexto de cada breakpoint no editor ou na barra de ferramentas da janela do breakpoint.



Opção 1: Contagem de acertos (Hit Count)
No exemplo, podemos usar o conhecimento de que isso ocorrerá na 257ª iteração.
(*Nos casos em que isso não é conhecido de antemão, consulte o parágrafo sobre "contagem de passes" para saber como obter esse número*)
Vamos definir um breakpoint no início de **`Hex2Int`** on line **`13`**. Em seguida, vamos para as propriedades e definimos "**Hit Count**" para **257**.



Isso ignorará as primeiras 256 ocorrências do ponto de interrupção, ignorando o cálculo dos valores de 0 a 255. A execução do aplicativo nos levará ao ponto de interrupção, e podemos avaliar "h" para ver que ele contém a string hexadecimal `100`.



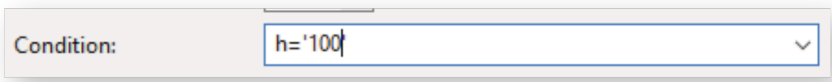
Now, we can fallback to the methods we have used in some of the previous articles.
Using stepping and watches we will find that it works fine for the lower 2 digits (*both zero*) at the end, but then multiplies the "1" with 265. A simple typographical error, easy to fix. It should be 256.

```
9.      d: array [0..3] of integer = (1, 16, 256, 4096);
```

After that, the application runs to the end, as it should.

NOTE: For backends other than **FpDebug** the hit-count may be zero or one-based. It may need to be adjusted accordingly. Hit-Count is handled directly by the backend (**gdb, lldb or FpDebug**).

Opção 2: Condition
Uma abordagem alternativa é usar uma condição no ponto de interrupção. Assim, não precisamos saber com que frequência ele pode ser atingido. Tudo o que precisamos ter é o valor de alguma variável ou expressão. No nosso caso, sabemos que "h" deve ser '100'. Inserimos isso nas propriedades:



A execução do aplicativo com esse ponto de interrupção nos levará à mesma chamada de "Hex2Int", com o parâmetro "h" = '100'.

OBSERVAÇÃO: As condições no ponto de interrupção podem funcionar de forma diferente dependendo do backend do depurador. Com o FpDebug, qualquer expressão Pascal deve funcionar. Com outros back-ends (gdb ou lldb), a condição precisa ser suportada por cada um deles.
A comparação de cadeias de caracteres não funciona com nenhum deles. Também dependendo do backend, se uma expressão não puder ser computada ou não retornar um valor booleano, o depurador poderá sempre parar (ignorar a expressão) ou nunca parar (tratá-la como falsa).
O **FpDebug** ignorará essas expressões.

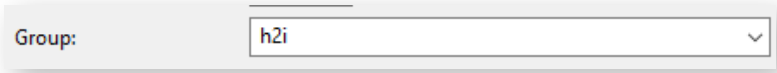
No nosso caso, tudo o que precisávamos era a condição. Chegamos ao código no estado em que estávamos interessados, no primeiro hit em que a condição era verdadeira. Em outros casos, as condições podem testar apenas alguns aspectos do estado necessário, e o breakpoint ainda pode parar algumas vezes antes que o erro seja alcançado. Para isso, as condições e a contagem de ocorrências podem ser combinadas.
No FpDebug, a contagem de ocorrências será aplicada somente às ocorrências para as quais a condição era verdadeira. Assim, você poderia verificar se o primeiro último em "h" era '0' definindo a condição: **h [3] = '0'** e **Hit-Count=17**, o que interromperia a 17ª vez que "h" terminasse com um '0'.

Opção 3: Deixar o autor da chamada decidir

Às vezes, não temos uma boa condição para testar quando o problema surgirá. Mas talvez saibamos que ele sempre acontece quando determinados critérios são atendidos no chamador ou em algum outro código que é executado antes de o bug se manifestar. Nesse caso, precisamos de um ponto de interrupção nesse outro código (neste exemplo, no chamador) para avaliar a condição.
Para isso, precisamos de dois pontos de interrupção. O primeiro estará na **linha 13**, como antes. No entanto, desta vez, definiremos esse ponto de interrupção como desativado.

```
begin
13  Result := 0;
    for i := 0 to Length(h)-1 do begin
```

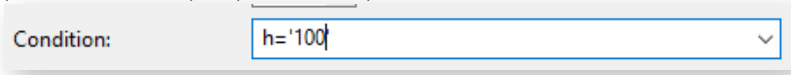
Esse ponto de interrupção não terá uma condição nem um Hit-Count. Mas nós o atribuiremos a um grupo. Usaremos "**h2i**" como nome do grupo, para que possamos identificá-lo facilmente como o grupo para a função "**Hex2Int**".



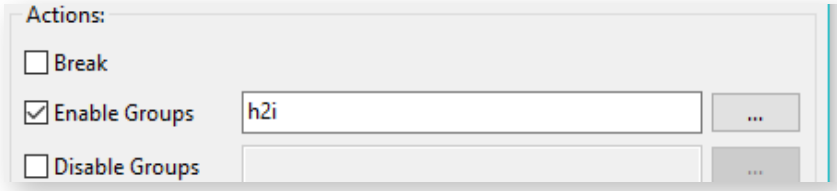
O segundo breakpoint vai para o chamador. Nós o usaremos para manter a condição e, depois, para ativar o primeiro breakpoint quando a condição for atendida. Dessa forma, o aplicativo acabará parando no primeiro breakpoint, mas podemos controlar quando isso ocorrerá.
No exemplo, colocamos esse ponto de interrupção na **linha 30**.

```
    t := t1+t2+t3;
30  i := Hex2Int(t);
    WriteLn('$'+t, ' = ', i);
```

Precisamos alterar algumas de suas propriedades. Primeiro, precisamos definir uma condição. Afinal, ele está no loop e não queremos que nada aconteça até que tenhamos os dados corretos para a chamada que queremos depurar.



E precisamos alterar o que acontece quando a condição desse breakpoint é atendida. Como já mencionado, queremos que ele ative o outro ponto de interrupção. Para isso, marcamos **"Enable Groups"** e inserimos o nome do grupo que usamos para o primeiro breakpoint.
Mas isso não é suficiente. Afinal, esse é um **breakpoint** e, assim que a condição for atendida, o aplicativo será pausado nele. Só que não queremos isso, queremos apenas pausar no outro **breakpoint**. Portanto, desmarcaremos a propriedade "**break**". Isso significa que o debugger nunca fará uma pausa aqui, ele só executará qualquer outra ação que o ponto de interrupção tenha.

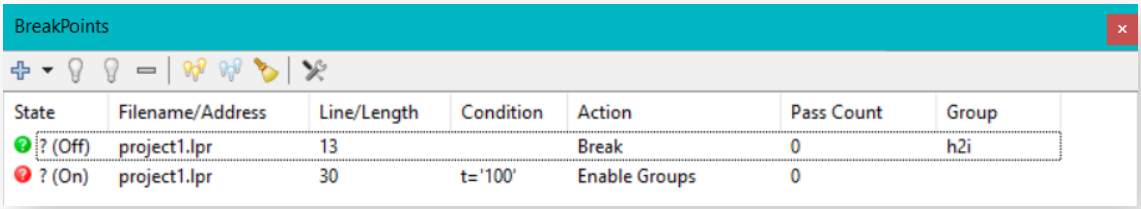


Com essa configuração, estamos prontos para executar o aplicativo. E, novamente, ele nos levará para "**Hex2Int**" e fará uma pausa quando os argumentos acionarem o bug.

OBSERVAÇÃO: em vez de usar uma condição no segundo ponto de interrupção, poderíamos ter usado o Hit-Count. No entanto, há um bug que impede que essa combinação funcione. Portanto, a partir do Lazarus 3.0, usar Hit-Count e Enable-Groups juntos ainda não funciona.

Mais propriedades do breakpoint

Como a oportunidade se apresenta, vamos dar uma olhada nas propriedades restantes dos pontos de interrupção. Além disso, faremos uma breve menção à janela do ponto de interrupção, que pode ser aberta no menu **View → Debug Windows → Breakpoints** ou usando **Ctrl-Alt-B**



Ele fornece uma visão geral de todos os seus pontos de interrupção e mostra algumas das propriedades que você definiu. A imagem mostra os dois breakpoints de "**Deixe o autor da chamada decidir**". Você pode ver o primeiro breakpoint tem um grupo. E o Segundo breakpoint tem o "**Enable Groups**", mas não a ação "**Break**".
No entanto, a visão geral não mostra quais grupos serão ativados. A janela também fornece botões de ferramentas para acessar a caixa de diálogo de propriedades, bem como para adicionar, remover, ativar e desativar breakpoints.

Contagem de acertos e contagem de passes

Na janela Ponto de interrupção, podemos ver a coluna "**Pass Count**". Trata-se de um simples contador da frequência com que o breakpoint foi atingido. Ele só conta quando o breakpoint está ativado. E, dependendo do backend, somente se a condição de interrupção for atendida.
Esse é o valor com o qual o **Hit-Count** configurado é comparado. O breakpoint agirá (ou seja, pausar, ou executar qualquer outra ação configurada) somente se a contagem de acertos for maior ou igual à contagem de aprovações. Com o **Pass Count,** você tem um contador simples da frequência com que uma linha é executada. Basta desmarcar a ação "break" e observar o Pass Count. No entanto, em linhas de código com frequência muito alta, até mesmo a simples contagem pode introduzir uma lentidão perceptível.
Como exemplo, obter essa contagem pode ser útil para o primeiro exemplo. Se não soubéssemos quantas vezes o ponto de interrupção precisa ser ultrapassado antes de ocorrer o erro, então poderíamos ter usado o Pass-Count.

"Break" e Auto Continue

A ação "Break" já foi usada no terceiro exemplo. Por padrão, ela está definida, e o ponto de interrupção será pausado. Se um ponto de interrupção for definido para executar outras ações, ele poderá ser desmarcado. Ela também pode ser desmarcada sem nenhuma outra ação, tornando o ponto de interrupção um simples contador. Outro efeito de desmarcar essa opção é que o IDE não assumirá o foco quando esse ponto de interrupção for atingido. Juntamente com "**Take Snapshot**" abaixo, isso pode ajudar na depuração de código sensível ao foco. "**Auto continue**" é semelhante, ele entrará em um estado de pausa, mas não transferirá o foco para o IDE. Ele aguardará o número especificado de milissegundos e, em seguida, continuará com "**run**" (*como se F9 tivesse sido pressionado*).
Como o "**Auto continue**" usa "run", ele não funcionará bem se você estiver passando por uma função, e atingir um ponto de interrupção do "**Auto Continue**" nessa função. Isso transformará o "**Step Over**" em "**Run**". Desmarcar a opção "**Break**" não interferirá no passo a passo.
Por outro lado, o "**Auto continue**" aguarda um momento. Assim, você pode dar uma olhada em Watches, Locals ou outras janelas que tenha aberto. E, durante o tempo de espera, você também pode optar por pressionar o botão de pausa, que interromperá o contador e pausará o aplicativo até que você mesmo o continue.

Ativar e desativar grupos

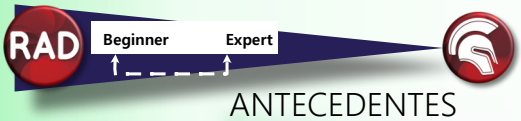
Retomando o exemplo anterior, há mais alguns detalhes de interesse. No exemplo, colocamos o ponto de interrupção com a opção "**Enable-Groups**" no chamador, mas isso não é obrigatório. Qualquer breakpoint pode ativar ou desativar qualquer outro. Portanto, sempre que seu código fizer algo que possa afetar outro código, você poderá ativar ou desativar os breakpoints nos locais afetados.

Na maioria das vezes, isso é feito como uma ativação única ou ativação e desativação de pontos de interrupção que existem em pares.
Outra opção útil é que um breakpoint pode se desativar. Isso pode ser útil se você tiver um breakpoint que registre algo ou tire um instantâneo. Você pode ativá-lo a partir de algum outro breakpoint e, em seguida, ele é atingido enquanto está ativado e se desativa. Assim, o registro em log só acontece uma vez após cada vez que o código passa por um dos pontos de interrupção que o habilitarão. Você pode ter mais de um ponto de interrupção em cada grupo, e um ponto de interrupção pode ativar ou desativar vários grupos. Os breakpoints que estão em um grupo podem, ao mesmo tempo, ativar ou desativar outros breakpoints. Assim, você pode configurar cadeias de pontos de interrupção habilitados. De modo geral, qualquer breakpoint em um grupo não é diferente de qualquer ponto de interrupção não agrupado. Sempre que é ativado, ele pode ter condições, Hit-Count e tudo o mais.

Eval Expression, Menssagem de Log, Log CallStack
Essas 3 ações registrarão dados no Event log. "**Eval Expression**" interpretará sua entrada como watch. "**Log Message**" registrará o texto literal.Você pode encontrar o registro de eventos no menu **View → Debug Windows → Event log** ou abra-o pressionando **Ctrl-Alt-V**. O registro de eventos mostra uma variedade de mensagens sobre o processo depurado. As mensagens que são mostradas podem ser decididas nas opções disponíveis no menu de contexto da janela. Para ver os dados de registro dos pontos de interrupção, é necessário ativar o **Breakpoints** na caixa de seleção das opções.

Take Snapshot
Essa opção adicionará um "**snapshot**" (*permanente*) no histórico de debug. Se um breakpoint não tiver essa opção, quando ele pausar o aplicativo ("**break**" habilitado), ele adicionará uma entrada não permanente. Com essa opção, a entrada será adicionada à lista de entradas permanentes. E com essa opção, mesmo um ponto de interrupção que tenha "break" desativado registrará um snapshot. Esse é um recurso importante se você precisar depurar algo que seja sensível ao foco ou algo que mude de comportamento se o IDE o interromper por muito tempo. (*Lembre-se de que, ao tirar um instantâneo, o aplicativo ainda será executado mais lentamente, pois o depurador precisa de algum tempo para tirar o instantâneo*).
Um **snapshot** contém: A Janela thread, a janela da pilha com os **cinco primeiros stack-frames**, o watches e a janela de locais para o stack frame superior do thread atual. Ele pode conter mais quadros de pilha, se a janela de pilha estiver aberta e definida para exibir mais quadros. Mas ainda assim, ele só receberá watches e locals para o quadro superior.Poderíamos ter depurado nosso problema inicial usando esse recurso. Se tivéssemos definido pontos de interrupção na função **`Hex2Int`** e coletado instantâneos sem pausar, assim que o aplicativo tivesse parado com o resultado errado, poderíamos ter examinado os instantâneos e visto o que aconteceu.
HISTÓRICO DE DEBUG
Sempre que o depurador pausa o aplicativo, ele exibe informações como a pilha, watches e locals, além de uma lista de threads que o aplicativo possui. O histórico de depuração mantém uma cópia dos dados das últimas 25 vezes em que o aplicativo foi pausado.
Essas entradas são listadas na janela de histórico, que pode ser acessada no menu **View → Debug Windows** → History ou via Ctrl-Alt-H. A seleção de uma entrada na janela fará com que a Watches, Locals, Stack e Thread exibe os valores dessa entrada em vez do valor atual.
Se precisar de uma entrada para sobreviver ao limite de 25 entradas, você pode pressionar o botão **Add Snapshot** botão. A entrada será mantida até que você a exclua ou termine a debug e inicie uma nova sessão de debug. A opção **`breakpoint`** " **Take snapshot** " também adiciona essa entrada permanente. Isso permite coletar dados e analisá-los posteriormente. Alguns exemplos de como isso pode ser útil são:

- **Debugging de aplicativos sensíveis ao foco**, ou aplicativos que tenham um gancho global de mouse ou teclado e que se comportariam de forma diferente se você enviasse uma entrada de tecla ou de mouse para o IDE.
- **Coleta de dados até que ocorra um erro.** Em seguida, para voltar e analisar o que aconteceu antes do erro.
- **Salvando os dados de debug** coletados e enviá-lo a um colega de trabalho para receber feedback. (*A janela do histórico de depuração tem uma função de importação/exportação*)
- **Ter outra pessoa executando seu aplicativo no debugger em seu sistema.** Você enviaria a eles um projeto, com breakpoints pré-configurados (*tirando instantâneos, mas não "interrompendo"*).
  E eles enviariam de volta o histórico coletado.



ANTECEDENTES

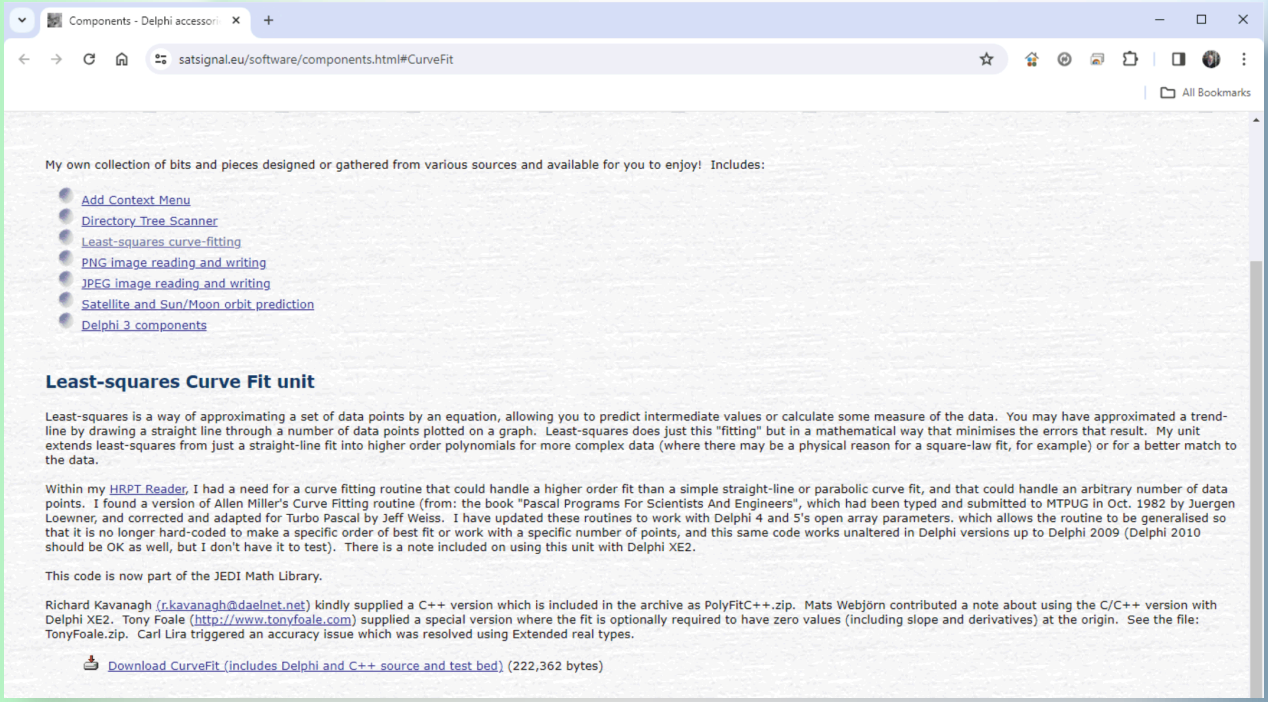
No campo da metalurgia de precisão, a capacidade de modelar e reproduzir com precisão formas e curvas complexas a partir de um conjunto de pontos de dados é crucial.
Esse desafio nos foi apresentado quando um cliente exigiu um software capaz de acionar uma máquina de usinagem para produzir curvas suaves e precisas com base em pontos amostrados. Ele tinha que produzir uma saída que fosse uma curva suave, com base em um conjunto de pontos amostrados. Tendo trabalhado em problemas semelhantes anteriormente, eu tinha uma ideia aproximada de como isso deveria funcionar matematicamente, mas, como geralmente acontece no desenvolvimento de software, eu tinha certeza de que outra pessoa teria uma ideia, eu tinha certeza de que outra pessoa já teria resolvido esse problema específico.

REQUISITOS

O objetivo principal era desenvolver um software capaz de gerar uma curva que se ajustasse perfeitamente a um determinado conjunto de pontos. Essa tarefa envolvia não apenas encontrar a curva de melhor ajuste, mas também aprimorar sua resolução.
Isso foi conseguido com a introdução de pontos adicionais, garantindo um alto grau de suavidade. Por exemplo, a transformação de uma entrada de 400 pontos em uma saída composta por 5.000 pontos que se encontram precisamente na curva calculada.
Esse requisito é fundamental em aplicações em que a precisão influencia diretamente a qualidade do produto final, como na usinagem de metais.

PESQUISA

Uma rápida pesquisa no Google identificou algumas possíveis soluções Delphi para esse problema. A que parecia mais promissora era a unidade de ajuste de curva de mínimos quadrados de David Taylor:
`https://www.satsignal.eu/software/components.html#CurveFit`
Essa era uma única unidade Delphi que não dependia de componentes de terceiros e simplesmente usava a unidade System.Math. A rotina foi escrita no Delphi 5, mas o projeto de teste foi compilado na versão mais recente do Delphi sem nenhum problema, como seria de se esperar de uma biblioteca puramente matemática.
O procedimento principal está listado na próxima página: Artigo Página 2/5



```
procedure PolyFit (const x, y: array of real;
                   var coefs: array of real;
                   var correl_coef: real;
                   const npoints, nterms: Integer);
var
  error: boolean;
  i, j: integer;
  xi, yi, yc, srs, sum_y,sum_y2: real;
  xmatr: matrix;      // Data matrix
  a: matrix;
  g: array of real;    // Constant vector

begin
  if nterms < 1 then
    Raise EMathError.Create ('PolyFit called with less than one term');
  if npoints < 2 then
    Raise EMathError.Create ('PolyFit called with less than two points');

  SetLength (g, nterms);
  SetLength (a, nterms, nterms);
  SetLength (xmatr, npoints, nterms);

  for i := 0 to npoints - 1 do
    begin     { setup x matrix }
      xi    := x [i];
      xmatr [i, 0] := 1.0;            { first column }
      for j  := 1 to nterms - 1 do
      xmatr [i, j] := xmatr [i, j - 1] * xi;
    end;

  square (xmatr, y, a, g, npoints, nterms);
  GaussJordan (a, g, coefs, nterms, error);
  sum_y    := 0.0;
  sum_y2   := 0.0;
  srs      := 0.0;

  for i := 0 to npoints - 1 do
    begin
      yi := y [i];
      yc := 0.0;
      for j := 0 to nterms - 1 do
        yc    := yc + coefs [j] * xmatr [i, j];
      srs     := srs + sqr (yc - yi);
      sum_y   := sum_y + yi;
      sum_y2  := sum_y2 + yi * yi
    end;

  // If all Y values are the same, avoid dividing by zero
  correl_coef := sum_y2 - sqr (sum_y) / npoints;
  // Either return 0 or the correct value of correlation coefficient
  if correl_coef <> 0 then correl_coef := srs / correl_coef;
  if correl_coef >= 1
    then correl_coef := 0.0
    else correl_coef := sqrt (1.0 - correl_coef);

  g := nil;
  a := nil;
  xmatr := nil;
end;
```

USANDO A BIBLIOTECA

A função principal da biblioteca selecionada exigia entradas na forma de matrizes de valores X e Y, juntamente com uma especificação do número de termos (grau do polinômio) a ser usado.
Esse aspecto é fundamental, pois determina a complexidade e a precisão da curva resultante. Um único termo geraria uma linha horizontal simples, enquanto o aumento dos termos introduz graus mais altos de polinômios, oferecendo um ajuste mais preciso ao custo da complexidade computacional.

ENTRADAS

A função principal aceita:

- Uma matriz de valores X de ponto flutuante.
- Uma matriz de valores Y de ponto flutuante.
- O número de termos a serem usados.

O número de termos está relacionado ao grau da fórmula polinomial que será usada.
Um único termo equivale a uma linha horizontal em um gráfico, por exemplo, e.g. $y = 2$
O uso de dois termos equivale a uma linha reta em um gráfico, por exemplo, e.g. $y = 3x + 2$
O uso de três termos equivale a uma equação quadrática, por exemplo, e.g. $y = 4x^2 + 3x + 2$

SAÍDAS

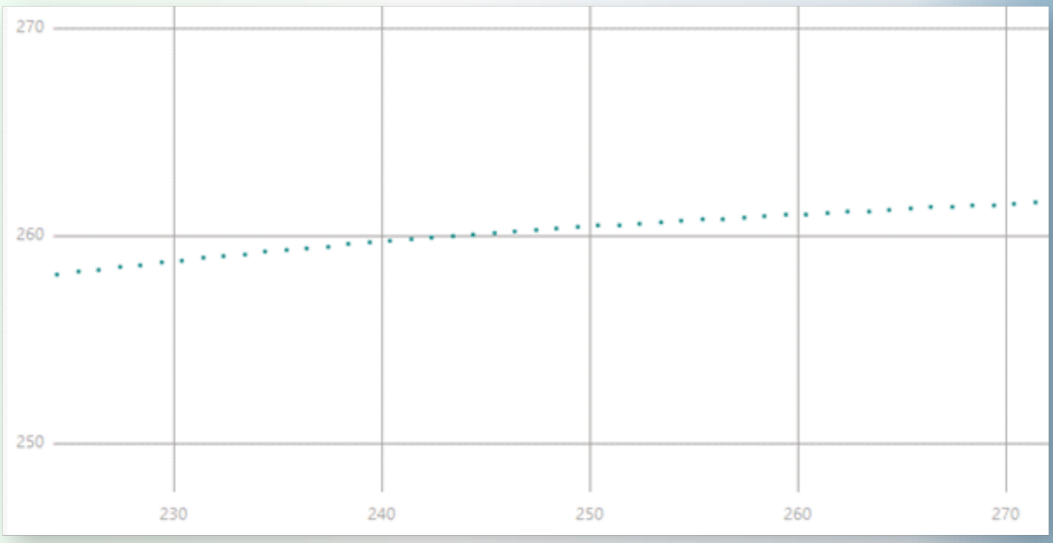
A função de cálculo preencherá uma matriz de valores de ponto flutuante que correspondem aos coeficientes da equação. Por exemplo, se quisermos usar três termos (uma equação quadrática), podemos passar uma matriz de três números de ponto flutuante. No exemplo quadrático acima, os coeficientes seriam 2, 3 e 4, respectivamente.
O cálculo também retorna um coeficiente de correlação, que nos diz o quanto a curva se correlaciona com os pontos de entrada fornecidos. Quanto mais próximo de 1 for esse coeficiente, mais preciso será o ajuste.

RESULTADOS

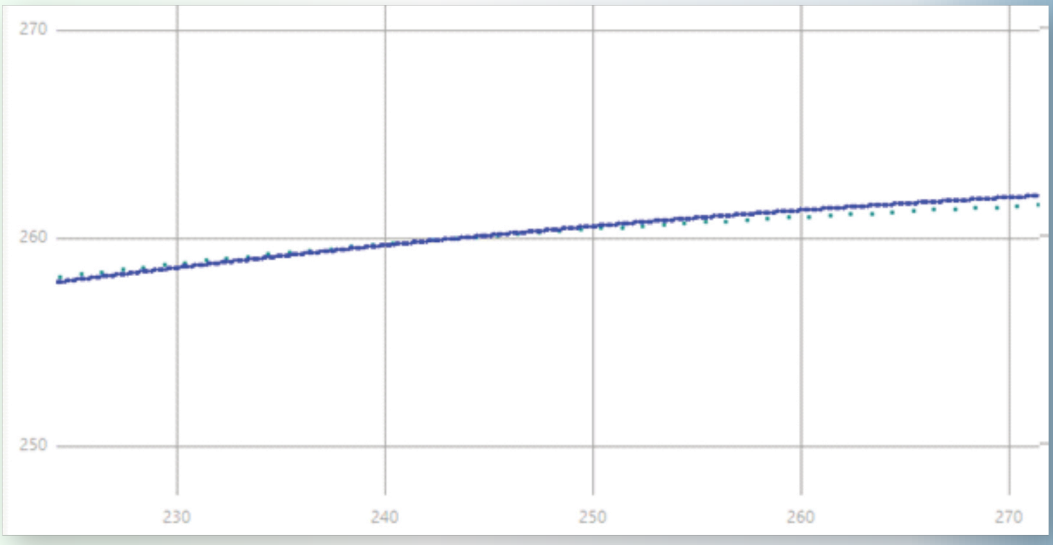
Ao usar a biblioteca, os testes unitários foram escritos usando um conjunto de pontos de teste fornecidos pelo nosso cliente. Os pontos fornecidos eram típicos das amostras que seriam usadas na produção. Com esses pontos de amostra, a curva mais precisa foi produzida ao chamar a função com seis termos. Isso equivale a uma função polinomial de quinto grau.
Ao usar mais de seis termos, ocasionalmente foram produzidos erros de estouro de ponto flutuante, provavelmente porque os coeficientes calculados eram muito pequenos para serem representados em um tipo Double no Delphi. Ao usar menos de seis termos, a rotina ainda calculava uma curva de melhor ajuste, mas, é claro, o ajuste não era tão preciso. Com menos termos, o coeficiente de correlação foi menor.
Com seis termos, recebemos seis coeficientes de volta na matriz fornecida à função.
Quando tivermos os coeficientes, poderemos calcular o valor Y para qualquer valor de X.
Para obter a curva desejada, foi calculada uma matriz de 5.000 pontos X uniformemente espaçados, delimitados pelos valores máximo e mínimo de X da amostra.

```
function CalculateY(const X: Double; const Coefficients: TArray<Double>): Double;
begin
  var yc := 0.0;
  var xc := 1.0;

  for var i := Low(Coefficients) to High(Coefficients) do
  begin
    yc := yc + Coefficients[i] * xc;
    xc := xc * X;
  end;

  Result := yc;
end;
```

Aqui está uma amostra de alguns dos pontos de entrada:



In the software, the resulting best-fit curve is calculated and overlaid over the points as follows:

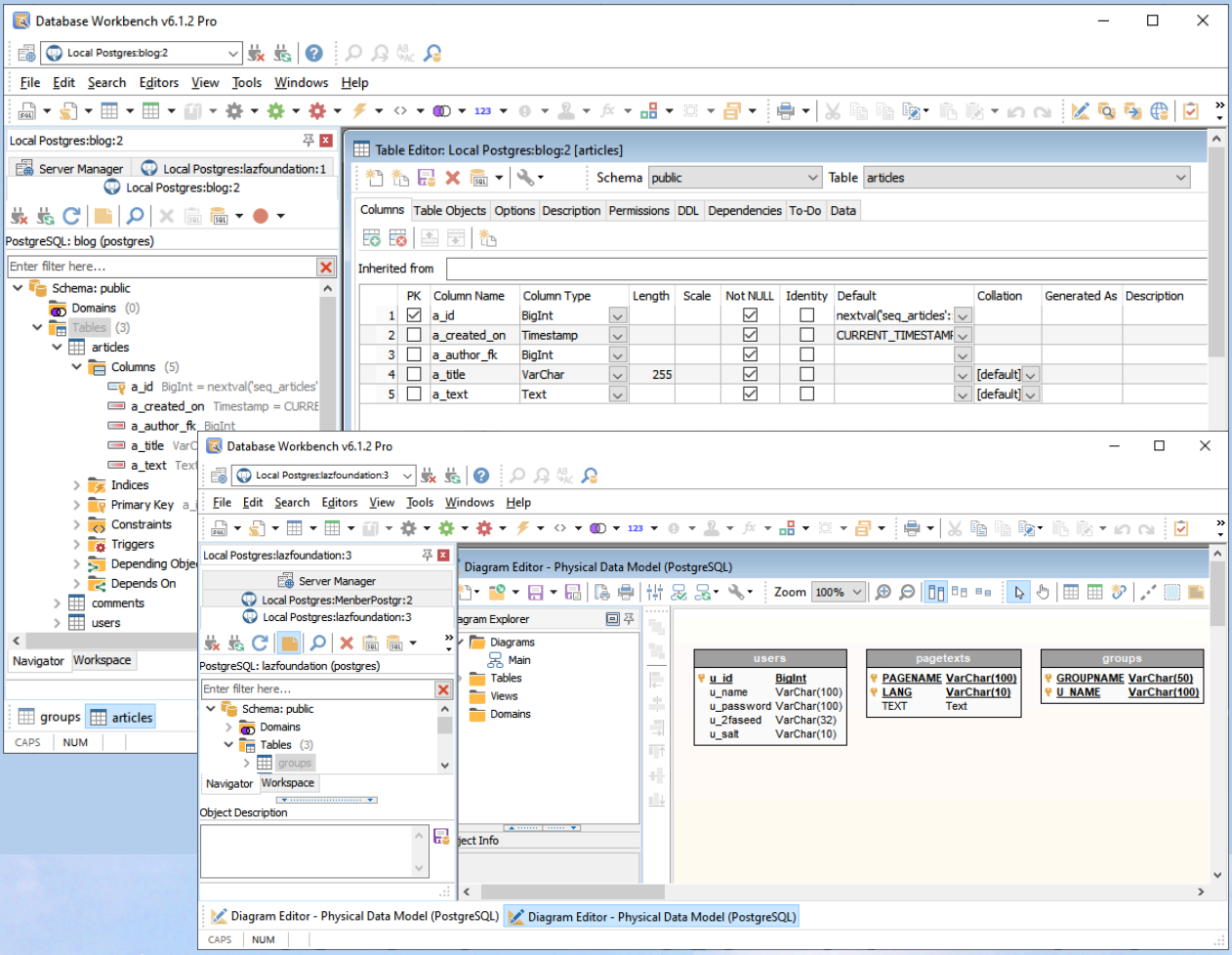


IMPLEMENTAÇÃO

A função usa um algoritmo de ajuste de curva de mínimos quadrados usando **Gauss-Jordan Elimination.** Esse é um algoritmo bem conhecido para encontrar uma curva polinomial de melhor ajuste para um conjunto de pontos.

CONCLUSÃO

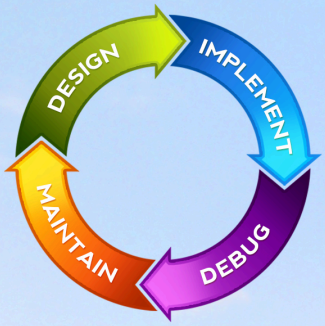
Vimos que uma biblioteca matemática originalmente escrita em **Turbo Pascal** e nas primeiras versões do **Delphi** pode ser usada nas versões mais recentes do **Delphi** sem modificações. Essa biblioteca pode produzir curvas precisas e suaves para o melhor ajuste de um conjunto de pontos de dados. Quanto maior o número de termos e, portanto, o grau do polinômio, mais preciso será o ajuste.



Introducing

Database Workbench 6

database development environment



Consistent user interface, modern code editors, Unicode enabled, HighDPI aware, ER designer, reverse engineering, meta data browsing, visual object editors, meta data migration, meta data compare, stored routine debugging, SQL plan visualizer, test data generator, meta data printing, data import and export, data pump, Grant Manager, DBA tasks, code snippets, SQL Insight, built in VCS, report editor, database meta data search, numerous productivity tools and much more...

for SQL Server, Oracle, MySQL, MariaDB, Firebird, InterBase, NexusDB and PostgreSQL

Upscene *Database tools for developers*

www.upscene.com

Starter Expert

RESUMO
O envio de uma mensagem a um smartphone ou a um aplicativo da Web faz parte de muitos aplicativos. Normalmente, essas mensagens são enviadas por um serviço em segundo plano.
O **Free Pascal** contém unidades com as quais você pode enviar **push notifications.**
Uma olhada mais de perto.

❶ INTRODUÇÃO

Há ocasiões em que você pode querer enviar uma mensagem para as pessoas da sua comunidade, ou aos seus usuários, para notificá-los sobre um evento interessante ou relevante ou sobre uma notícia.
Para o navegador, há um padrão chamado '**WebPush**' que permite enviar mensagens para um navegador.

O navegador mantém em execução um processo em segundo plano (*os chamados service workers*) que escuta essas mensagens e as exibe quando elas chegam.
Os smartphones têm sua própria versão do protocolo de mensagens: dependendo do sistema operacional (*Android ou iOS*), a **API** é diferente.

Independentemente da plataforma, o modo de operação é o mesmo: o processo começa pedindo permissão ao usuário para mostrar-lhe notificações.
Depois que a permissão é concedida, o sistema operacional ou o navegador pode se inscrever no serviço de **mensagens push**, que retorna um token exclusivo que pode ser usado para enviar mensagens ao dispositivo no qual o usuário concedeu permissão. O token é válido até que o usuário retire sua permissão.

O Google oferece um serviço que permite que você use esse token para enviar uma mensagem, independentemente da plataforma que emitiu o token: **Firebase Cloud Messaging (FCM)**.
Ele permite o envio de mensagens e fornece estatísticas sobre como os usuários reagiram às suas mensagens. Esse serviço pode ser acessado usando uma API REST simples.
Graças a um generoso patrocínio, o **Free Pascal** contém uma unidade que permite que você use essa API sem a necessidade de se preocupar com os detalhes da API. A seguir, examinaremos essa unidade.

❷ PRÉ-REQUISITOS

Antes de nos aprofundarmos na API, é necessário fazer alguns preparativos. O FCM não é um serviço gratuito e, se você planeja enviar muitas mensagens, precisará pagar.
Não é de surpreender que o Google exija que você registre o aplicativo que deseja usar para enviar mensagens. Se ainda não tiver feito isso, será necessário começar criando um projeto do Firebase no **console do Google Firebase:** (*Consulte a página 2 deste artigo*)
`https://console.firebase.google.com/`

❷ PRÉ-REQUISITOS

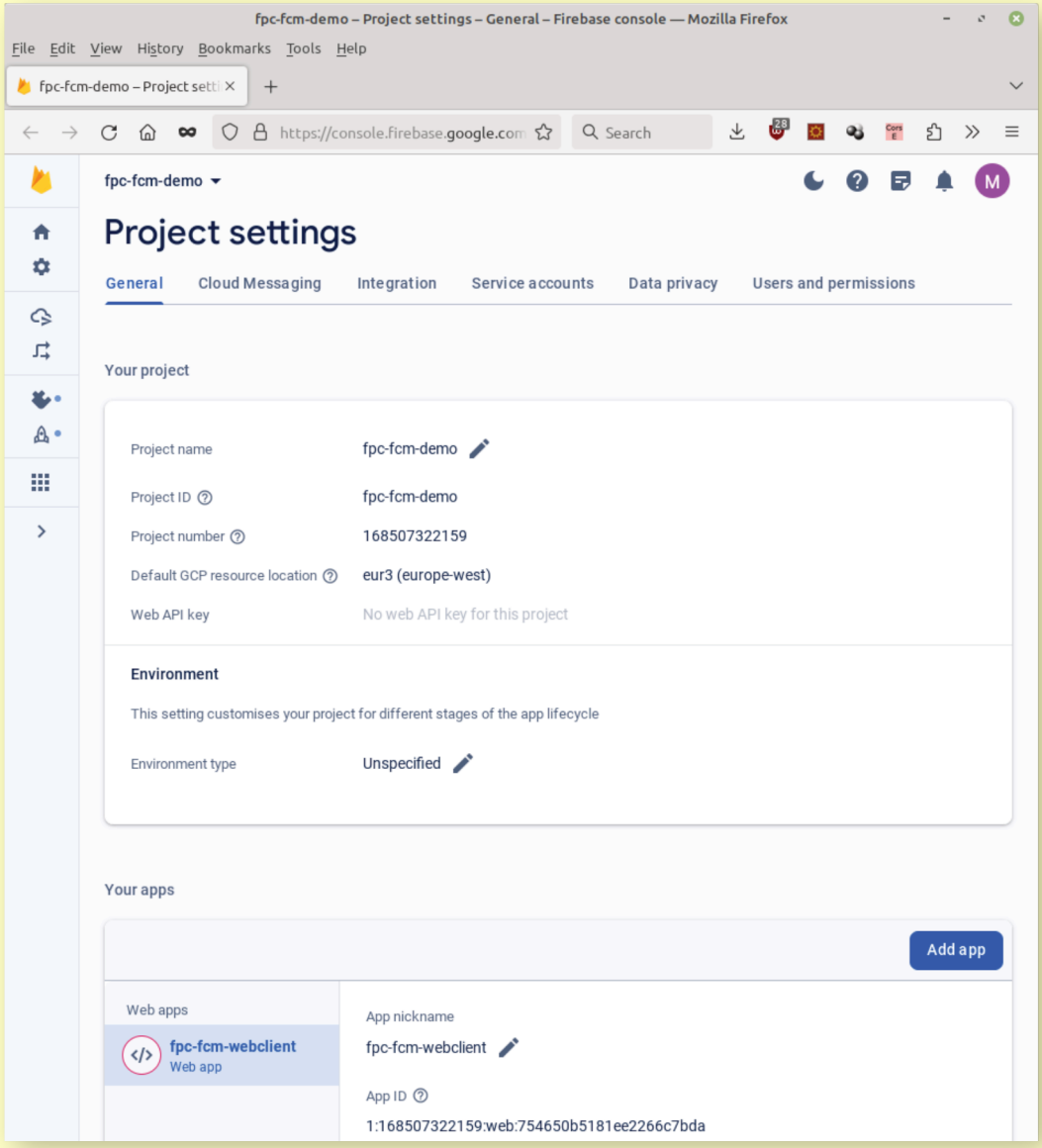


Figura 1: Um projeto finalizado do Firebase

É claro que você precisará ter uma conta do Google para fazer login nesse serviço. Qualquer conta válida do Google pode ser usada. O menu suspenso do projeto contém um item de menu para criar um projeto.

Um passo a passo completo da criação de um projeto está fora do escopo deste artigo, mas o processo é simples e está bem documentado em outro lugar (embora os detalhes das telas mudem regularmente). Quando terminar, você deverá ter algo parecido com a *figura 1 na página 2* deste artigo. A próxima etapa é registrar o aplicativo que acessará o serviço **Google Firebase**. Vários aplicativos podem usar o mesmo projeto do **Firebase Cloud Messaging**, e você deve registrá-los separadamente:

Na página mostrada na *figura 1 na página 2 deste artigo*, você pode ver um botão com o qual pode criar um aplicativo.

❷ PRÉ-REQUISITOS

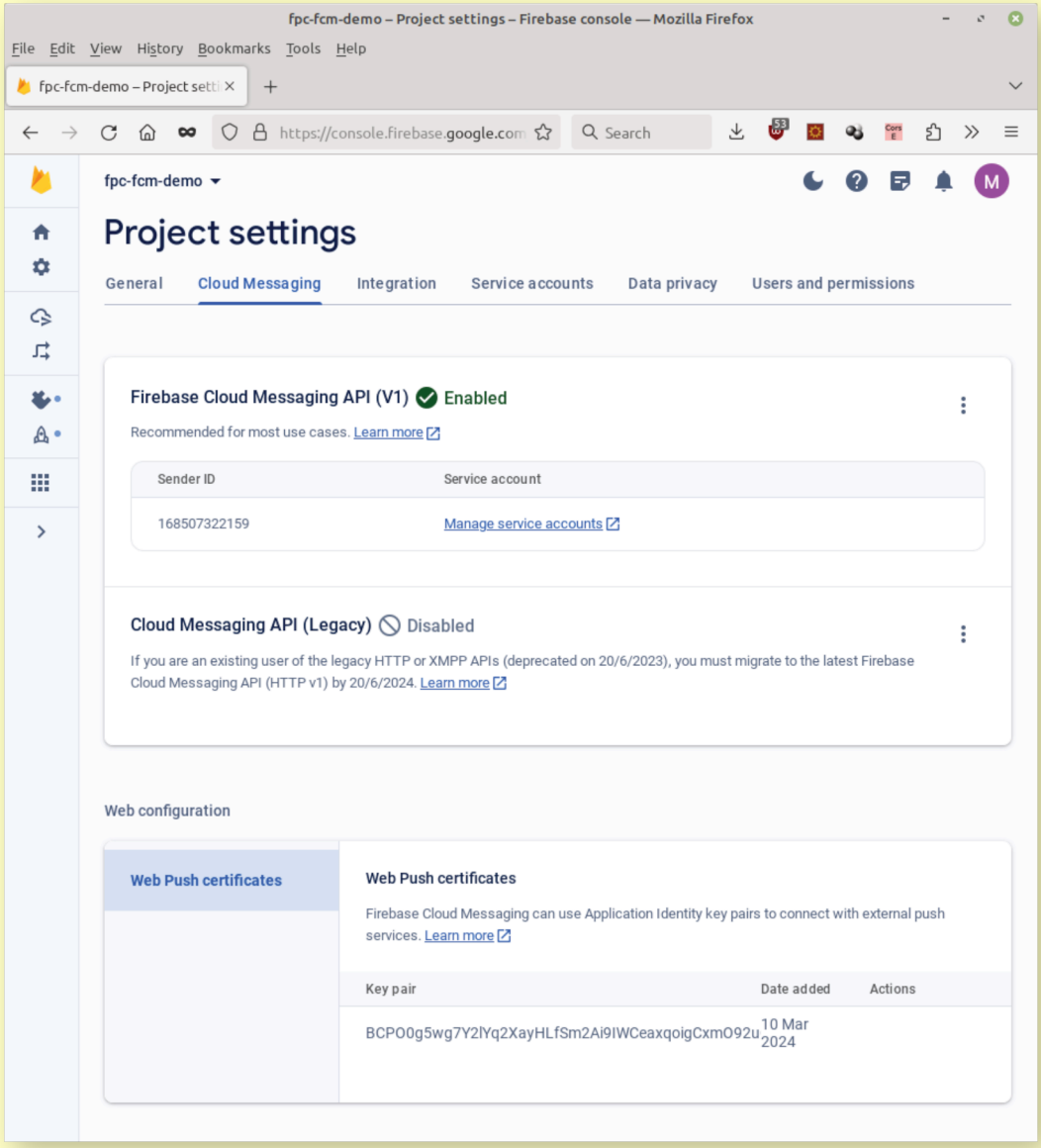


Figura 2: Página do Cloud Messaging com a chave VAPID

A definição do aplicativo é importante: quando o aplicativo é criado, você pode fazer download de um arquivo de configuração de aplicativo **JSON** que precisa ser usado ao **acessar as APIs do Google FCM:**
Entre outras coisas, ele contém um identificador de projeto exclusivo.
O conteúdo do arquivo será semelhante a este:

```
{ apiKey: "XYZ",
  authDomain: "fpc-fcm-demo.firebaseapp.com",
  projectId: "fpc-fcm-demo",
  storageBucket: "fpc-fcm-demo.appspot.com",
  messagingSenderId: "123",
  appId: "1:123" }
```

❷ PRÉ-REQUISITOS

Se desejar enviar mensagens aos usuários no navegador, você precisará de uma chave **VAPID**. Essa é uma chave especial que identifica o aplicativo do navegador e é exigida pelo protocolo **WebPush** usado pelo Firebase para enviar mensagens ao navegador.
Quando o projeto **Firebase** for criado, você poderá obter uma chave **VAPID** na página do projeto, na página "Cloud Messaging" (*consulte a Figura 2 na página 3*), abaixo de "**Web Push Certificates**". A chave pública precisa ser copiada, pois terá de ser usada no navegador.
A última etapa é a criação de uma conta de serviço:
A conta de serviço é necessária para enviar mensagens. Essa conta é usada para obter um token de acesso para autenticar as solicitações às APIs REST de envio HTTP do Firebase. Você pode obter a chave privada da conta de serviço na página "Contas de serviço" do seu projeto, mostrada na figura 3 na página 5 deste artigo.
Assim como o arquivo de configuração de informações do aplicativo, você precisa fazer o download do arquivo de informações da conta de serviço e salvá-lo em algum lugar do seu disco rígido. Ele deve se parecer com o seguinte:

```
{ "type": "service_account",
   "project_id"    : "fpc-fcm-demo",
   "private_key_id": "123456789",
   "private_key"   : "XXXYYYZZZ",
   "client_email"  : "firebase-adminsdk@gserviceaccount.com",
   "client_id"     : "987654321",
   "auth_uri"      : "https://accounts.google.com/o/oauth2/auth",
   "token_uri"     : "https://oauth2.googleapis.com/token",
   "auth_provider_x509_cert_url":
      "https://www.googleapis.com/oauth2/v1/certs",
   "client_x509_cert_url": "https://www.googleapis.com/",
   "universe_domain": "googleapis.com" }
```

Quando todas essas etapas forem concluídas, você terá todas as informações para começar.

❷ O OBJETO DE MENSAGEM

Enviar uma mensagem para as APIs do **Firebase Cloud Messaging** significa enviar um objeto **JSON**. A página a seguir descreve a mensagem que pode ser enviada:

https://firebase.google.com/docs/reference/fcm/rest/v1/projects.messages#resource:-message

A unidade fpfcmtypes contém uma definição de classe **Object Pascal** que oferece a você todos os campos presentes na especificação. Isso tem a vantagem de permitir que você use o recurso de autocompletar código.

```
TNotificationMessage = class(TJSONPersist)
public
  procedure ToJSON(aObj : TJSONObject);
  function Encode : string;
  procedure Clear;
  // toplevel properties, valid for all platforms.
  // Notification data.
  Property Recipient : String;
  Property RecipientType : TRecipientType;
  property Data: TStrings;
  property Title: string;
  property Body: string;
  property Image: string;
  // available in Apple and Android, not in web.
  Property Options: TMessageOptions;
  Property SendOptions : TNotificationSendOptions;
  // Apple specific
  Property AppleConfig : TAppleConfig;
// Android specific
  Property AndroidConfig : TAndroidConfig;
// Web specific
  Property WebPushConfig : TWebPushConfig;
// FCM options
  Property FCMOptions : TFCMOptions;
end;
```

❸ A MENSAGEM OBJETO

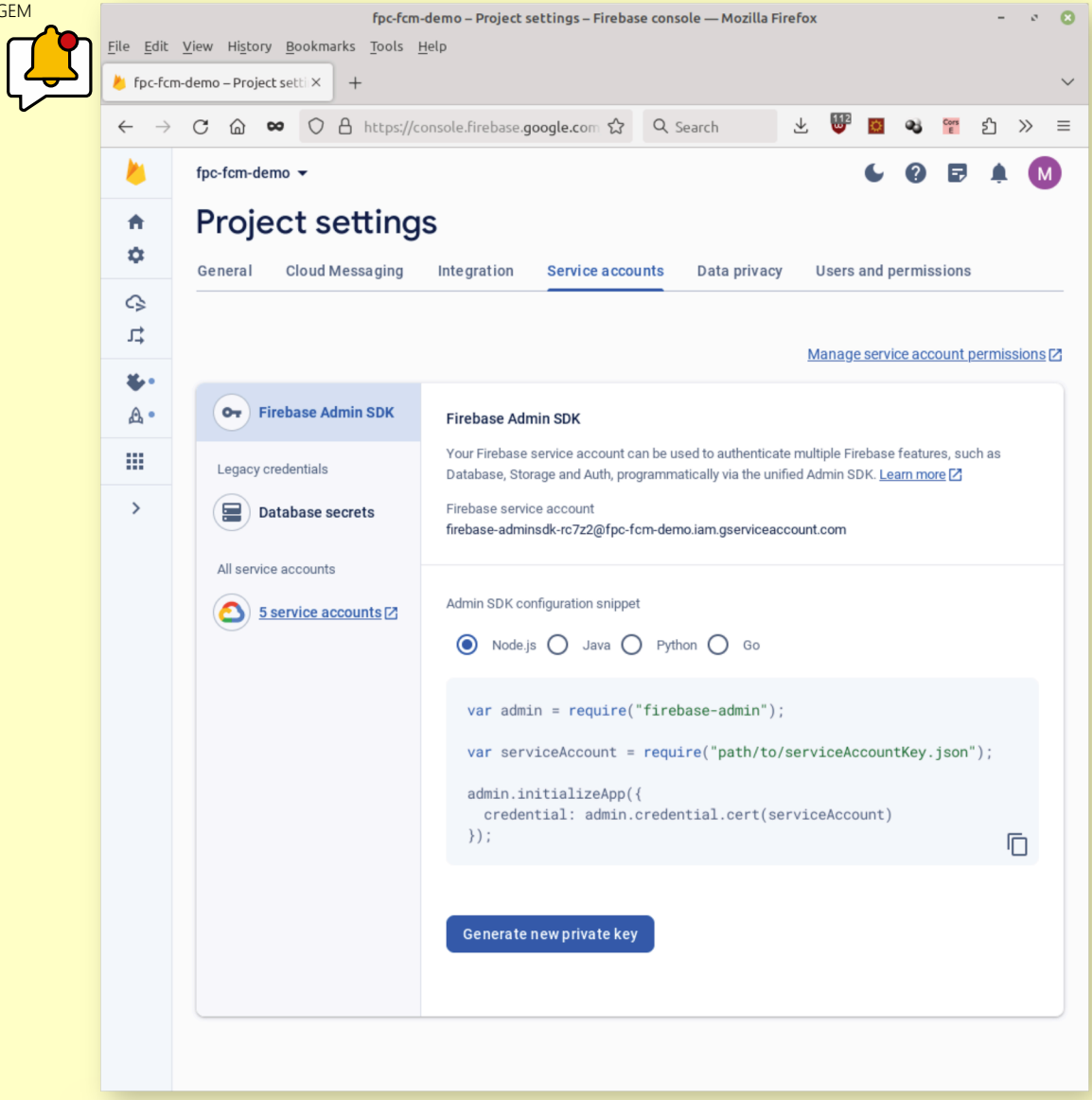


Figura 3: A página de informações da conta de serviço

O significado das propriedades Recipient, Title, body e Image deve ser intuitivamente claro.
A propriedade `RecipientType` é do tipo `TRecipientType` e pode ser uma das seguintes `rtToken`, `rtTopic`, `rtCondition`: você pode enviar mensagens para uma pessoa ou para várias pessoas.
AppleConfig, AndroidConfig, WebPushConfig e FCMOptions contêm opções de configuração específicas que o **FCM** usará ao enviar a mensagem para uma dessas plataformas.
No uso diário, você provavelmente não precisará defini-las, mas todas elas estão detalhadas na página da Web acima:
As propriedades desses objetos refletem as propriedades especificadas na especificação.

❸ A MENSAGEM OBJETO

A propriedade SendOptions é um conjunto que informa ao objeto de mensagem quais dessas partes de configuração opcionais ele deve incluir na mensagem **JSON**.
Depois que você tiver preenchido as propriedades do objeto, a função encode retornará uma cadeia de caracteres que contém a mensagem **JSON** a ser enviada ao servidor **FCM**. O método Clear limpará todas as propriedades. Portanto, para enviar a mesma mensagem a vários usuários, preencha as propriedades da mensagem, e, em um loop, define o destinatário, recupera o **JSON** e envia o **JSON** para o servidor **FCM**.

❹ THE CLIENT OBJECT

A unidade `fpfcmclient` contém o objeto `TFCMClient`. Essa classe é o cliente usados para se comunicar com o servidor FCM. Ela cuida da autenticação e do envio da mensagem. A classe não tem muitos métodos ou propriedades

```
TFCMClient = class(TComponent)
Public
  Procedure InitServiceAccount(const aFileName: string;
                                                  aRoot: TJSONStringType);
  Procedure InitServiceAccount(const aJSON : TJSONObject);
  function Send(aMsg : TNotificationMessage; aRecipient : UTF8String) : Boolean;
  function Send(aMsg : TNotificationMessage; aRecipients :
                                                  Array of UTF8String) : Boolean;
  Property WebClient : TAbstractWebClient;
  Property BearerToken: TBearerToken;
  Property ServiceAccount: TServiceAccountData;
  Property LogFile: String;
  Property OnError: TFCMErrorEvent;
  Property OnNewBearerToken : TFCMBearerTokenEvent;
  Property OnResponse: TFCMResponseEvent;
end;
```

As propriedades são bastante simples:

WebClient

Isso pode ser definido como uma instância de uma classe abstrata `TWebClient`.
Essa classe abstrai os detalhes do protocolo HTTP e mostraremos como usá-la.
Se você não o definir, a classe `TFCMClient` criará uma instância padrão.

BearerToken

O token de autenticação do portador a ser usado na próxima solicitação HTTP. Você pode definir isso se o tiver armazenado em algum lugar. Se nenhum tiver sido definido ou se o token tiver expirado, um token de portador será obtido conforme necessário usando os detalhes da conta de serviço

ServiceAccount

Essa propriedade fornece acesso somente leitura aos dados da conta de serviço.
Essa propriedade deve ser inicializada com o método InitServiceAccount.

LogFile

Defina essa propriedade como o nome de um arquivo se quiser um registro das comunicações HTTP com os servidores do Google FCM: todos os cabeçalhos e corpos de solicitação e resposta serão gravados nesse arquivo.

Os eventos a seguir são totalmente opcionais:

OnError

Você pode definir esse evento se quiser tratar os erros por conta própria. Se não for definido, será gerada uma exceção em caso de erro.

OnNewBearerToken

Como mencionado anteriormente, um token de portador será obtido conforme necessário usando os detalhes da conta de serviço. Esse evento é chamado quando um novo token de acesso é recebido. Você pode usá-lo para armazená-lo para a próxima chamada.

OnResponse

Esse evento é chamado com a resposta de envio: você pode armazenar a resposta em um banco de dados, se desejar.

❹ THE CLIENT OBJECT
O OBJETO CLIENTE

A classe tem apenas 2 métodos públicos:
`InitServiceAccount` Você pode chamá-lo com o nome de um arquivo e um caminho opcional para um elemento raiz. Isso inicializará os dados da **ServiceAccount**. O arquivo a ser fornecido é o arquivo de configuração da conta de serviço que você baixou ao configurar a conta de serviço. Você também pode obter os dados **JSON** por outros meios e fornecer diretamente os dados **JSON**.
Enviar Isso enviará a mensagem para o servidor **FCM** usando a **API HTTP REST**. Você pode especificar um ou vários destinatários. O campo **Recipient** (Destinatário) da mensagem será preenchido com cada um dos destinatários especificados, e a mensagem será enviada. A função retornará True se a mensagem tiver sido enviada com êxito *(se você não tratar os erros usando OnError, uma exceção será levantada quando algo der errado*).
Com essa classe em mãos, como podemos enviar uma notificação por push? De forma bem simples. Para demonstrar isso, criamos um aplicativo de console baseado em TCustomApplication. Em seu método DoRun, inserimos o seguinte código:

```
Recip :=GetOptionValue('r','recipient');
Msg   :=nil;
Client:=TFCMClient.Create(Self);
Try
  ConfigureClient(Client);
  Msg  :=TNotificationMessage.Create;
  ConfigureMessage(Msg);
  Client.Send(Msg,Recip);
Finally
  Msg.Free;
  Client.Free;
end;
```

A maior parte do trabalho ocorre nas chamadas **ConfigureClient** e **ConfigureMessage**:
Para configurar o cliente, precisamos de vários itens, todos os quais serão fornecidos por meio das opções de linha de comando do nosso programa.
opções de linha de comando do nosso programa. A primeira coisa a fazer é inicializar a conta de serviço. O arquivo com as informações da conta de serviço pode ser especificado usando a opção de linha de comando **-s**, mas um nome de arquivo padrão será usado se não for especificado:

```
procedure TFCMApplication.ConfigureClient(aClient : TFCMClient);

const Err = 'No service account configuration file found: %s';

var CfgFile : string;

begin
  // Service account info
  CfgFile:=GetOptionValue('s','service-account');
  if CfgFile='' then
  CfgFile:=ChangeFileExt(ParamStr(0),'-service-account.json');
  if not FileExists(CfgFile) then
  Raise EInOutError.CreateFmt(Err,[CfgFile]);
  aClient.InitServiceAccount(CfgFile,'');
  // Access token reuse
  if HasOption('a','access-token') then
  begin
    FAccessTokenFile:=GetOptionValue('a','access-token');
    // Load initial token
    if FileExists(FAccessTokenFile) then
      aClient.BearerToken.LoadFromFile(FAccessTokenFile);
    // Set handler so we save the token when it was fetched.
    aClient.OnNewBearerToken:=@DoHandleNewToken;
    end;
  // Log file
  if HasOption('l','log') then
    aClient.LogFile:=GetOptionValue('l','log');
end;
```

❹ THE CLIENT OBJECT
O OBJETO CLIENTE

Usando a opção **-a** você pode especificar o token do portador:
O registro do token do portador tem um método para carregar o token de um arquivo ou para salvá-lo em um arquivo. Normalmente, um token tem uma vida útil de aproximadamente 1 hora, após a qual um novo token deve ser obtido. Por fim, o arquivo de registro pode ser definido usando a opção **-l.**
O método `DoHandleNewToken` é chamado quando um novo token é solicitado. Nesse método, o token pode ser salvo em um arquivo:

```
procedure TFCMApplication.DoHandleNewToken(Sender: TObject;
                                           const aToken: TBearerToken);
begin
  aToken.SaveToFile(FAccessTokenFile);
end;
```

O método `ConfigureMessage` define as propriedades da mensagem de notificação.
Você pode especificar um arquivo **JSON** para carregar a mensagem (usando a opção **-m**).
Ou você pode especificar o corpo da mensagem, o título e a imagem com as opções **-b, -t -i,** respectivamente.

```
procedure TFCMApplication.ConfigureMessage(Msg : TNotificationMessage);
begin
  if HasOption('m','message') then
    LoadMessageFromFile(Msg,GetOptionValue('m','message'));
  if HasOption('t','title') then
    Msg.Title :=GetoptionValue('t','title');
  if HasOption('b','body') then
    Msg.Body  :=GetoptionValue('b','body');
  if HasOption('i','image') then
    Msg.Body  :=GetoptionValue('i','image');
end;
```

O `LoadMessageFromFile` é novamente bastante simples:

```
procedure TFCMApplication.LoadMessageFromFile(Msg : TNotificationMessage;
    const aFileName : string);
Var
  F   : TFileStream;
  D   : TJSONData;
  Obj: TJSONObject absolute D;

begin
  D:=Nil;
  F:=TFileStream.Create(aFileName,fmOpenRead or fmShareDenyWrite);
  try
    D:=GetJSON(F);
    if not (D is TJSONObject) then
      Raise EFCM.CreateFmt('Invalid JSON data in message file %s',[aFileName]);
    Msg.Title :=Obj.Get('title',Msg.Title);
    Msg.Body  :=Obj.Get('body',Msg.Body);
    Msg.Image :=Obj.Get('image',Msg.Image);
  finally
    D.Free;
    F.Free;
  end;
end;
```

Como você pode ver, o arquivo de mensagem é um arquivo **JSON** simples com 3 chaves: título, corpo e imagem.
Com isso, o cliente está quase pronto para ser usado. Há um pequeno detalhe a ser observado: a propriedade **WebClient** não está definida em lugar algum. A classe `TFCMClient` criará então um cliente Web padrão. O cliente Web padrão precisa ser configurado, e isso pode ser feito adicionando as seguintes unidades à cláusula uses:

❹ THE CLIENT OBJECT
O OBJETO CLIENTE

fphttpwebclient
isso define o cliente da Web padrão como uma classe baseada em `TFPHTTPClient`;

opensslsockets
Isso habilita o suporte a HTTPS para o `TFPHTTPClient`. O HTTPS é necessário para a comunicação com os serviços FCM.

Além disso, no código de início do programa, precisamos adicionar a seguinte linha:
`DefaultWebClientClass:=TFPHTTPWebClient;`
Isso instruirá a classe `TFCMClient` a usar a classe `TFPHTTPWebClient` ao criar uma instância do `TWebClient`.

Com esse código no lugar (*além de algum código auxiliar para mostrar uma mensagem de uso e algumas verificações das opções de linha de comando*), o programa pode ser executado a partir da linha de comando da seguinte forma:
`sendmsg -m message.json -s service.json -a token.json -r TOKEN`
Aqui, **TOKEN** precisa ser substituído por um token obtido ao pedir permissão ao usuário para enviar uma mensagem a ele. Os dados da conta de serviço serão carregados do arquivo service.json (*que você deve ter baixado usando o console do projeto* **FCM**).
A mensagem é especificada no arquivo message.json:

```
{
"title" : "A nice message",
"body" : "With a nice body",
"image" : "https://www.freepascal.org/favicon.png"
}
```

Se você especificar o comando várias vezes com diferentes tokens de destinatário, verá que o token é salvo no arquivo token.json e reutilizado para as próximas chamadas.

❺ OBTENÇÃO DE UM TOKEN USANDO UM SITE

No exemplo acima, não mostramos como obter um token para enviar uma mensagem a um usuário; presumimos que ele estivesse disponível. Na prática, esse token deve ser obtido do usuário, solicitando sua permissão para enviar mensagens. Essa é uma função que está disponível nos sistemas operacionais móveis e no navegador.
O site usaremos o **PAS2JS** para demonstrar como obter esse token e usá-lo para enviar uma mensagem ao navegador. O navegador tem uma interface simples para mostrar notificações, que é - sem surpresa - chamada **Notification** (Notificação) e que é detalhada aqui:
`https://developer.mozilla.org/en-US/docs/Web/API/notification`
O método para solicitar permissão para mostrar notificações é - Pode ser mais simples? - chamada `requestPermission`.
Quando chamado, o navegador exibirá uma mensagem solicitando sua permissão para mostrar suas notificações. O valor de retorno é uma **Promis**, que será preenchida quando o usuário tiver concedido (*ou negado*) a permissão. Depois de obter a permissão, você pode obter o token que pode ser usado para enviar mensagens. Para fazer isso, o gerenciador Push pode ser usado:
`https://developer.mozilla.org/en-US/docs/Web/API/PushManager`
A chamada de assinatura resultará em uma assinatura, que pode ser convertida em um token.
Os desenvolvedores do **Firebase** criaram uma pequena camada de **API** em torno dessa chamada, e seguiremos suas diretrizes para obter e usar um token. Essa é uma precaução simples: na verdade, a documentação do **Firebase** não deixa claro se o **Firebase** simplesmente usa o token bruto obtido do navegador ou acrescenta algumas informações necessárias para entregar a mensagem.
A API de mensagens do **Firebase** Cloud foi disponibilizada na unidade firebaseapp.

Nós a usaremos para obter um token e enviaremos esse token para um pequeno aplicativo de servidor HTTP, que o usará para enviar uma notificação Push usando o componente `TFCMClient` apresentado acima.

❺ OBTENÇÃO DE UM TOKEN USANDO UM SITE

Portanto, para esse fim, criamos um projeto "Aplicativo de navegador da Web" no Lazarus. Isso criará uma página HTML e um arquivo de programa. Na página HTML, alguns arquivos precisam ser adicionados para que seja possível usar a API do Firebase:

```
<script src="firebase-app-compat.js"></script>
<script src="firebase-messaging-compat.js"></script>
```

Esses arquivos podem ser baixados do site do Firebase, conforme descrito abaixo:

https://firebase.google.com/docs/web/alt-setup

Estamos usando a camada de compatibilidade (porque é isso que está descrito na unidade firebaseapp),
mas também seria possível usar a API modular.
Para inicializar um aplicativo Firebase, é necessário usar o arquivo de configuração do aplicativo que você criou e baixou quando definiu seu projeto no console do Firebase.
Você pode incluir a definição do objeto JSON no código do aplicativo, mas também pode colocá-lo em um arquivo no seu servidor HTTP:

```
var firebaseConfig = {
authDomain: "fpc-fcm-demo.firebaseapp.com",
projectId: "fpc-fcm-demo",
storageBucket: "fpc-fcm-demo.appspot.com",
messagingSenderId: "123",
appId: "1:123"
};
```

Em seguida, você pode incluir esse arquivo (vamos chamá-lo de config.js) no arquivo HTML principal do seu projeto junto com o include do arquivo do projeto:

```
<script src="config.js"></script>
<script src="webclient.js"></script>
```

O restante do HTML é bastante simples:
adicionamos 1 controle de edição (edtMessage) e 2 botões (btnSend e btnRegister) ao HTML, além de algumas tags DIV para exibir o token e algumas saídas do programa.

```
<div class="container">
  <div class="box">
    <h3 class="title is-3">FCM Push notification demo</h3>
    <div class="field">
      <label class="label">Message to send:</label>
    <div class="control">
      <input id="edtMessage" class="input" type="text" placeholder="Message to send">
    </div>
  </div> <!-- .field -->

  <div class="field is-grouped">
    <div class="control">
      <button id="btnSend" class="button is-link" disabled>Send</button>
    </div>
      <div class="control">
        <button id="btnRegister" class="button is-link is-light">Register</button>
      </div>
    </div> <!-- .field -->
  </div> <!-- .box -->
  <div id="pnlToken" class="box is-hidden">
    <p>Your token: <span id="lblToken">?</span> <p>
  </div> <!-- .box -->
</div> <!-- .container -->
<script>
rtl.run();
</script>
<div id="pasjsconsole"></div>
```

O elemento pasjsconsole é onde o feedback do WriteLn será exibido.
O código do aplicativo está em uma classe TDemoApp, um descendente da classe TBrowserApplication que vem por padrão com o Pas2JS. Quando o aplicativo é iniciado, o método DoRun é chamado. Nele, o código a seguir é executado para inicializar alguns campos que representam os vários elementos HTML div e os botões. Para esses últimos, ele também define os retornos de chamada de clique:

```
var config : TJSObject; external name 'firebaseConfig';

procedure TDemoApp.DoRun;

begin
  RPCModule   :=TRPCModule.Create(Self);
  pnlToken    :=GetHTMLElement('pnlToken');
  lblToken    :=GetHTMLElement('lblToken');
  edtMessage  :=TJSHTMLInputElement(GetHTMLElement('edtMessage'));
  btnSend     :=TJSHTMLButtonElement(GetHTMLElement('btnSend'));
  btnSend.addEventListener('click',@HandleSend);
  btnRegister :=TJSHTMLButtonElement(GetHTMLElement('btnRegister'));
  btnRegister.addEventListener('click',@HandleRegister);
  Writeln('Initializing application...');
  App:=Firebase.initializeApp(config);
  App.messaging.onMessage(@HandleReceivedMessage);
  RegisterServiceWorker;
end;
```

As últimas linhas inicializam a API do Firebase usando o objeto config do nosso arquivo config.js e definem o manipulador de eventos OnMessage da API de mensagens do Firebird. O evento OnMessage pode ser usado para reagir a mensagens. Aqui, apenas exibiremos a mensagem de notificação criando uma instância TJSNotification:

```
procedure TDemoApp.HandleReceivedMessage(aMessage: TJSObject);
var
  Notif: TJSObject;
  Opts : TJSNotificationOptions;
begin
  if assigned(aMessage) then
    console.debug('Message received: ',aMessage);
  Notif      :=TJSObject(aMessage['notification']);
  Opts       :=TJSNotificationOptions.new;
  Opts.body  :=string(Notif['body']);
  Opts.image :=string(Notif['image']);
  TJSNotification.new(string(Notif['title']),opts);
end;
```

Podemos fazer isso porque solicitamos permissão do usuário para exibir notificações.
Observe que, quando a página da Web não estiver carregada e focada, o service worker exibirá a mensagem (também usando a API de notificação do navegador).
A última etapa ao inicializar o aplicativo é registrar um script de service worker:

```
procedure TDemoApp.RegisterServiceWorker;
begin
 Window.Navigator.serviceWorker.register('firebase-messaging-sw.js').
  _then(function (js : JSValue) :JSValue
    begin
      reg:=weborworker.TJSServiceWorkerRegistration(js);
      if assigned(Reg) then
        Writeln('Registered service worker...')
    end,function (js : JSValue) :JSValue
    begin
      Writeln('Unable to register service worker')
    end);
end;
```

No código acima, o script do service worker é chamado de "firebase-messaging-sw.js" e pode ser baixado das páginas de documentação do Firebase. O método de registro retorna uma promessa, que é resolvida para um objeto de registro do service worker. Aqui, apenas salvamos o registro no campo reg para uso posterior e exibimos uma mensagem para mostrar se o service worker foi registrado com sucesso ou não. Para páginas da Web mais elaboradas, é possível fazer mais coisas depois que o service worker for registrado, como estabelecer um canal de mensagens entre o service worker e a página principal da Web. Para o nosso exemplo atual, isso não é necessário. Um service worker é um serviço pequeno, mantido pelo navegador: Ele será executado em segundo plano e uma das coisas que pode fazer é ouvir as mensagens que chegam - exatamente o que ele precisa fazer para a nossa demonstração.
O script do service worker não faz muita coisa, exceto inicializar o aplicativo de mensagens do Firebase:

❺ OBTAINING A TOKEN USING A WEBSITE

```
importScripts('https://www.gstatic.com/firebasejs/9.2.0/firebase-app-compat.js');
importScripts('https://www.gstatic.com/firebasejs/9.2.0/firebase-messaging-compat.js');
importScripts('config.js');
firebase.initializeApp(firebaseConfig);
```

Com tudo isso, o aplicativo do navegador está pronto para receber mensagens push.
Para obter um token de mensagens do Firebase, o usuário deve clicar no botão "Register" (Registrar). No manipulador de "clique" do botão de registro (HandleRegister), o registro do service worker que salvamos anteriormente é passado para a chamada getToken da API de mensagens do Firebase para obter um token de mensagens do Firebase.

```
TFirebaseMessaging = class external name 'firebase.messaging.Messaging' (TJSObject)
function getToken (options : TMessagingGetTokenOptions): string; async;
end;
```

O objeto TMessagingGetTokenOptions tem um campo serviceworkerRegistration:
Quando definido, a API do Firebase sabe qual service worker estará lidando com o recebimento de mensagens. Além disso, o campo vapidkey deve ser definido como a chave VAPID que você criou quando o projeto Firebase foi criado. A constante VAPIDKey (não mostrada aqui) contém essa chave.

```
procedure TDemoApp.HandleRegister(event: TJSEvent);
var
  Token : string;
  opt : TMessagingGetTokenOptions;

begin
  opt:=TMessagingGetTokenOptions.New;
  opt.serviceworkerRegistration:=self.Reg;
  opt.vapidKey:=TheVAPIDKey;
  Token:=Await(App.messaging.getToken(opt));
  if (token='') then
    RequestPermission
  else
    HaveToken(token);
end;
```

A chamada GetToken é uma chamada assíncrona, portanto, retorna uma promessa. Usando o recurso Await, podemos transformá-la em uma string de token real. Se a cadeia de token estiver vazia, a API do Firebase ainda não tem um token e devemos solicitar ao usuário sua permissão para mostrar notificações. Se um token não vazio for retornado, poderemos enviá-lo ao servidor. Isso é feito usando as chamadas RequestPermission e HaveToken, respectivamente:

```
procedure TDemoApp.requestPermission;
  function onpermission (permission : jsvalue) : jsvalue;
  var
  token : string;
  begin
    if (permission='granted') then
      begin
        writeln('Notification permission granted.');
        handleregister(nil);
      end;
    end;
begin
  Writeln('Requesting permission...');
  TJSNotification.requestPermission()._then(@OnPermission)
end;
```

O método da classe RequestPermission do TJSNotification é um método do navegador.
Como a solicitação de permissão pode demorar um pouco, o resultado da chamada é uma promessa: Quando o usuário reage à solicitação de permissão para enviar mensagens, a promessa se transforma em uma cadeia de caracteres que contém a decisão do usuário: a permissão é concedida ou negada.
Quando a permissão é concedida, o método HandleRegister é novamente chamado: nesse caso, o método getToken da API de mensagens do Firebase será bem-sucedido e o token será enviado para a chamada HaveToken. Essa chamada mostra o token no HTML e envia o token para o nosso servidor de aplicativos:

❺ OBTAINING A TOKEN USING A WEBSITE

```
procedure TDemoApp.HaveToken(aToken : string);
begin
  Showtoken(aToken);
  Sendtoken(aToken);
  btnSend.disabled:=False;
  btnRegister.disabled:=False;
end;
```

O ShowToken é fácil, ele define o texto interno do elemento DIV com o ID lblToken:

```
procedure TDemoApp.ShowToken(aToken : string);
begin
  pnlToken.classlist.remove('is-hidden');
  lblToken.innerText:=aToken;
  Writeln('Received token: ',aToken);
end;
```

O SendToken usa uma chamada JSON-RPC RegisterSubscription para enviar o token ao nosso servidor de aplicativos:

```
procedure TDemoApp.SendToken(aToken : string);

  procedure DoOK(aResult: JSValue);
  begin
    Writeln('Registered token on server');
  end;

  procedure DoFail(Sender: TObject; const aError: TRPCError);
  begin
    Writeln('Failed to register token on server: '+aError.Message);
  end;

  begin
    Writeln('Sending token to server: ',aToken);
    RPCModule.Service.RegisterSubscription(aToken,@DoOK,@DoFail);
  end;
```

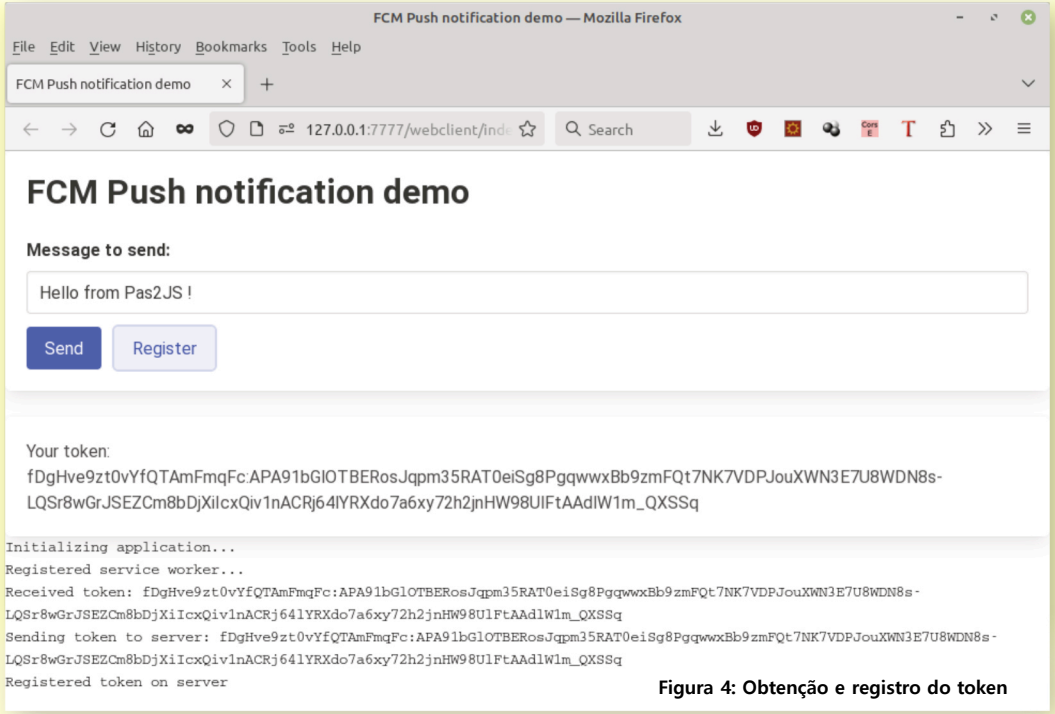


Figura 4: Obtenção e registro do token

O resultado desse código pode ser visto na Figura 4 na página 14 deste artigo.
Uma vez que o token tenha sido obtido e enviado com sucesso ao servidor, o usuário pode usar o botão Send (Enviar) para enviar uma mensagem para si mesmo. Em um aplicativo real, as mensagens serão, obviamente, enviadas pelo servidor em resposta a algum evento externo. O manipulador de eventos do botão Send é o HandleSend: Esse método constrói um pequeno objeto JSON com os dados da mensagem

❺ OBTAINING A TOKEN USING A WEBSITE

OBSERVE que o formato do objeto de mensagem é o mesmo que usamos para enviar uma mensagem usando o utilitário de linha de comando apresentado anteriormente. Quando o objeto de mensagem é construído, o manipulador usa a chamada SendNotification JSON-RPC para enviar a mensagem ao nosso servidor de aplicativos:

```
procedure TDemoApp.handlesend(event: TJSEvent);

  procedure DoOK(aResult: JSValue);
  begin
    Writeln('Message transferred to server for sending');
  end;

  procedure DoFail(Sender: TObject; const aError: TRPCError);
  begin
    Writeln('Failed to transfer message to server for sending: '+aError.Message);
  end;

var Msg : TJSObject;

begin
  Msg:=New([
    'title','Free Pascal FCM demo',
    'body',edtMessage.Value,
    'image','https://www.freepascal.org/favicon.png'
  ]);
  Writeln('Sending message : ',TJSJSON.stringify(Msg));
  RPCModule.Service.SendNotification(Msg,@DoOK,@DoFail);
end;
```

Isso conclui a parte interativa do aplicativo.

O servidor de aplicativos expõe um pequeno serviço JSON-RPC. Conforme mostrado em artigos anteriores sobre o PAS2JS e seu suporte para JSON-RPC, uma unidade com o objeto proxy para esse servidor JSON-RPC pode ser gerada automaticamente (a unidade é chamada service.messagingserver). O objeto proxy gerado tem a seguinte declaração:

```
TMessagingService = Class(TRPCCustomService)
Protected
  Function RPCClassName : string; override;
Public
  Function SendNotification (Message : TJSObject;
                             aOnSuccess : TJSValueResultHandler = Nil;
                             aOnFailure : TRPCFailureCallBack = Nil) : NativeInt;
  Function RegisterSubscription (Token : String;
                             aOnSuccess : TJSValueResultHandler = Nil;
                             aOnFailure : TRPCFailureCallBack = Nil) : NativeInt;
end;
```

Uma instância desse objeto proxy é criada em um módulo de dados RPCModule, que é implementado na unidade module.messagingservice:

```
TRPCModule = class(TDataModule)
 Client: TPas2jsRPCClient;
 procedure DataModuleCreate(Sender: TObject);
private
 FService: TMessagingService;
public
 Property Service : TMessagingService Read FService;
end;
```

O módulo de dados tem um componente TPas2JSRPCClient nele, e o objeto proxy é criado no evento OnCreate do módulo de dados RPCModule e conectado ao objeto RPCClient:

```
procedure TRPCModule.DataModuleCreate(Sender: TObject);
begin
  FService:=TMessagingService.Create(Self);
  FService.RPCClient:=Client;
end;
```

Com isso, nosso aplicativo Web está concluído.

❻ ENVIAR UMA MENSAGEM DE UM SERVIDOR DE APLICATIVOS

No exemplo acima, criamos uma página da Web configurada para receber e exibir notificações por push.
As notificações reais são enviadas por um servidor de aplicativos HTTP:
O servidor de aplicativos de demonstração é um aplicativo HTTP simples. Ele expõe um serviço JSON-RPC, que é implementado em uma unidade module.rpc. O tratamento real das chamadas RPC é implementado na unidade module.messaging, por 2 objetos TJSONRPCHandler chamados RegisterSubscription e SendNotification.
O manipulador de eventos OnExecute do objeto RegisterSubscription é bastante simples: ele extrai o token dos dados JSON passados pelo navegador

```
procedure TdmMessaging.RegisterSubscriptionExecute(Sender: TObject;
    const Params: TJSONData;
    out Res: TJSONData);
var
  Parms: TJSONArray absolute params;
  aToken : UTF8String;

begin
  If Parms.Count<>1 then
    Raise Exception.Create('Invalid param count');
  If Parms[0].JSONType<>JTString then
    Raise Exception.Create('Invalid param type for token');
  aToken:=Parms[0].AsString
  SaveToken(aToken);
  Res:=TJSONBoolean.Create(True);
end;
```

Quando a chamada ao método Savetoken retorna, um resultado é enviado de volta ao navegador. O método SaveToken usa uma lista de strings simples, que é carregada do arquivo, adiciona o token e salva novamente no arquivo:

```
procedure TdmMessaging.SaveToken(const aToken : UTF8String);
var
 L : TStrings;
 FN : String;
begin
 FN:=DeviceTokensFileName;
 L:=TStringList.Create;
 try
   if FileExists(FN) then
     L.LoadFromFile(FN);
   L.Add(aToken);
   L.SaveToFile(FN);
 finally
   L.Free;
 end;
end;
```

A função DeviceTokensFileName retorna o nome do arquivo no qual os tokens devem ser salvos.
devem ser salvos; os detalhes dessa função estão no código-fonte deste aplicativo.
O tratamento da mensagem SendNotification faz algo semelhante: extrai os dados do objeto JSON passado pelo cliente para preencher o objeto TNotificationMessage
objeto:

❼ ENVIO DE UMA MENSAGEM DE UM APLICATIVO SERVIDOR

```
procedure TdmMessaging.SendNotificationExecute(Sender: TObject;
                const Params: TJSONData;
                out Res: TJSONData);
var
 Parms : TJSONArray absolute params;
 Obj   : TJSONObject;
 Msg   : TNotificationMessage;

begin
 If Parms.Count<>1 then
   Raise Exception.Create('Invalid param count');
 If Parms[0].JSONType<>jtObject then
   Raise Exception.Create('Invalid notification');
 Obj:=Parms.Objects[0];
 Msg:=TNotificationMessage.Create;
 try
   Msg.Title :=Obj.Get('title',Msg.Title);
   Msg.Body  :=Obj.Get('body',Msg.Body);
   Msg.Image :=Obj.Get('image',Msg.Image);
   SendMessage(Msg);
   Res:=TJSONBoolean.Create(True);
 finally
   Msg.Free;
 end;
end;
```

Quando o objeto de mensagem é preenchido com os dados da página da Web, ele é passado para o método SendMessage, e um resultado é enviado de volta ao navegador.
O método SendMessage faz o trabalho real de enviar a mensagem de notificação por push com a classe TFCMClient. Ele começa carregando o último token do arquivo de token criado pela chamada RegisterSubscription:

```
procedure TdmMessaging.SendMessage(Msg : TNotificationmessage);
var
  Sender : TFCMClient;
  aConfig, aToken : String;

begin
  aToken:=LoadLastToken;
  Sender:=TFCMClient.Create(Self);
  try
    aConfig:=GetServiceAccountFileName;
    Sender.LogFile:=GetLogFileName;
    Sender.InitServiceAccount(aConfig,'');
    Sender.OnNewBearerToken:=@HandleNewAccessToken;
    if FileExists(AccessTokenFile) then
      Sender.BearerToken.LoadFromFile(AccessTokenFile);
    Sender.Send(Msg,aToken);
  finally
    Sender.Free;
  end;
end;
```

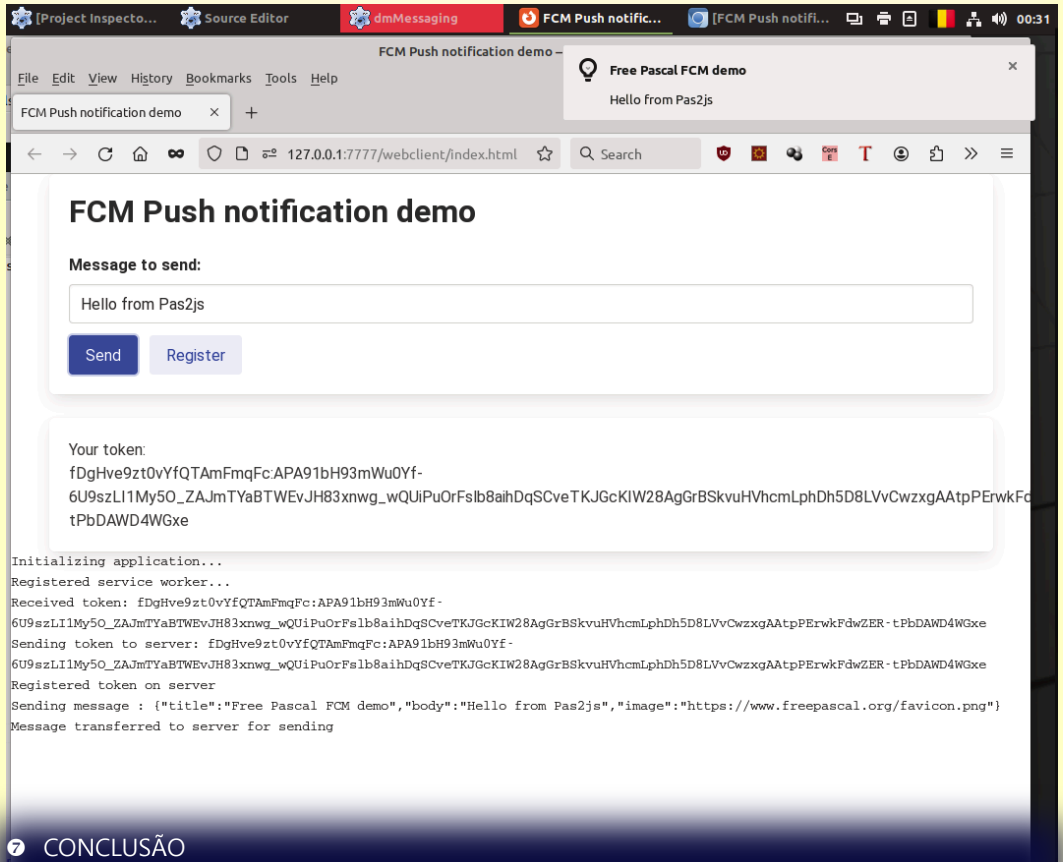
O método `HandleNewAccessToken` (usado para salvar o token de acesso) é idêntico ao do nosso utilitário de linha de comando.
Tudo o que resta a fazer é adicionar as unidades que definem a classe WebClient padrão a ser usada.
Depois disso, nosso aplicativo está concluído.
Deve ficar claro que, além de salvar e carregar o token de um arquivo, esse código do servidor de aplicativos não é substancialmente diferente do código do utilitário de linha de comando que construímos anteriormente.

❻ SENDING A MESSAGE FROM AN APPLICATION SERVER

Para executar esse exemplo, você deve seguir as etapas a seguir:

- Certifique-se de que o arquivo da conta de serviço esteja localizado próximo ao programa do servidor HTTP (**messageserver**) com o nome correto: **`messagserver-serviceaccount.json.`**
- Inicie o programa do servidor HTTP messageserver.
- Certifique-se de que o arquivo config.js esteja configurado corretamente conforme descrito acima, bem como os arquivos **Javascript** do os arquivos Javascript do firebase.
- Inicie o programa cliente baseado na Web no navegador.
  Para fazer isso, por exemplo, no **Lazarus**, basta pressionar **F9** para executá-lo,
  e ele deverá abrir no navegador.
- No navegador Firefox, certifique-se de que o console de ferramentas de desenvolvimento esteja aberto (**pressione F12**) e defina a opção ' **Enable service workers over HTTP** (*when toolbox is open*)' nas configurações da caixa de ferramentas.
- Essa etapa é necessária apenas para testes: Se você hospedar a página em um site HTTPS, os service workers serão automaticamente permitidos.
- Pressione o botão de registro no site. Você deverá ver uma mensagem de confirmação, como na *figura 4 da página 16 deste artigo.*
- Digite uma mensagem agradável e pressione " **Send** ".

O resultado deve se parecer com a *figura 5 na página 17*. Observe que, devido ao gerenciador de área de trabalho específico do autor, o ícone não é mostrado na imagem.
Se você anotou o token, também poderá enviar uma mensagem para o mesmo navegador usando a ferramenta de linha de comando.



❼ CONCLUSÃO

Neste artigo, mostramos que enviar notificações push a um usuário não é tão difícil de fazer. Na verdade, gasta-se mais tempo na definição de todos os arquivos necessários e na configuração do projeto no Firebase do que na codificação do aplicativo. O envio de mensagens fez uso do FCM - Firebase Cloud Messaging. É possível fazer o mesmo sem o Firebase, usando o protocolo WebPush embutido no navegador. Nos deixe a discussão sobre isso para uma contribuição futura.

A sessão de abertura foi iniciada com as saudações de Dieter Kranzmüller (BADW-LRZ), Uwe Heitmann do DLR-PT e Jorge Gasos e Stergios Tsiafoulis da DG CONNETC. O OpenWebSearch.EU teve início com uma reunião do consórcio em Berlim. 46 participantes se reuniram em um formato de reunião híbrida para trocar ideias e fazer planos para a execução bem-sucedida do projeto. A reunião foi organizada pelo DLR-PT em um novo prédio em Berlin-Südkreuz.

Open WebSearch .eu

NGI OPENWEBSEARCH.EU POR DETLEF OVERBEEK

14 renomados institutos europeus de computação e pesquisa uniram forças para criar uma infraestrutura aberta de pesquisa na Internet na Europa.
O projeto contribui para a **soberania digital da Europa** e pode proteger os valores de liberdade do mundo - especialmente na Europa - e promover um mercado de mecanismos de busca aberto e centrado nas pessoas.

Em setembro de 2022, parceiros, incluindo centros de computação e universidades, lançaram o projeto da **EU OpenWebSearch.eu**. Esse é o primeiro projeto financiado pela União Europeia para dar início ao futuro mecanismo de pesquisa na Web.

Nesse momento, foi lançada oficialmente a iniciativa **OpenWebSearch.eu**, financiada pela UE para viabilizar a pesquisa na Web. As preocupações com o desequilíbrio no mercado de mecanismos de busca formaram a base do projeto:
Google's índices da Web, como a página inicial, ou Bing's Páginas iniciais, usadas por DuckDuckGo, Ecosia, MetaGer, Neeva, Qwant and You.com.

Na prática, os resultados das respostas são limitados a esses nomes e, portanto, o mundo fica incompleto.Isso aumenta a desconfiança na sociedade.
Além dos dois índices de pesquisa dos EUA, existem apenas dois outros provedores comparáveis em todo o mundo após a aquisição do Yahoo pelo Bing (Microsoft).
a aquisição do Yahoo pelo Bing (Microsoft), que são o Yandex da Rússia e o Baidu da China.
Os dois últimos são claramente menos importantes para a Europa, principalmente por causa de suas diferenças de idioma e à apresentação pouco clara dos fatos.

A informação como um bem público, com acesso livre, imparcial e transparente, não está mais sob controle público. Essa falta de um ambiente de pesquisa aberto coloca em risco nossa liberdade e empobrece o poder inovador das sociedades, da tecnologia, dos institutos de pesquisa e da economia.

OBJETIVO

Nos próximos anos, os pesquisadores buscarão desenvolver o núcleo de um European Open Web Index (OWI) como base para um novo mecanismo de busca na Internet na Europa. Ele também estabelecerá as bases para um sistema europeu aberto e extensível de pesquisa e análise na **Web**. **Infrastructure** (**OWSAI**), com base em valores, princípios, legislação e padrões europeus.

- Portanto, poderia ser um grande - **livremente acessível ao público - Página inicial**.
  Na prática, parece que o público usa os sites principalmente como mecanismos de pesquisa.
  Que é o que eles eram originalmente: mecanismos de busca.
- **Assim, não haverá mais o Efeito de Manipulação do Mecanismo de Busca**:
- **O medo da manipulação (em larga escala) de dados e realidades** (**fake news**),
  transformando resultados e relatórios em desinformação ou falsas verdades.
  A formação de opinião pode ser influenciada por isso, o que serve apenas aos interesses de ditadores e antidemocráticos. E isso, por sua vez, afeta a democracia e nossa liberdade.
  Muito importante, portanto.

A intenção é desenvolver um índice da web contendo metade de todos os textos publicados na Internet.
Os parceiros envolvidos esperam um requisito de armazenamento de cerca de cinco petabytes (cinco milhões de gigabytes).
Em comparação com os índices do Google ou do Bing, esse é um banco de dados menor.
Cada um dos concorrentes estabelecidos tem centenas de petabytes de textos, arquivos de imagem, multimídia, dados de uso e arquivos de registro da Internet.

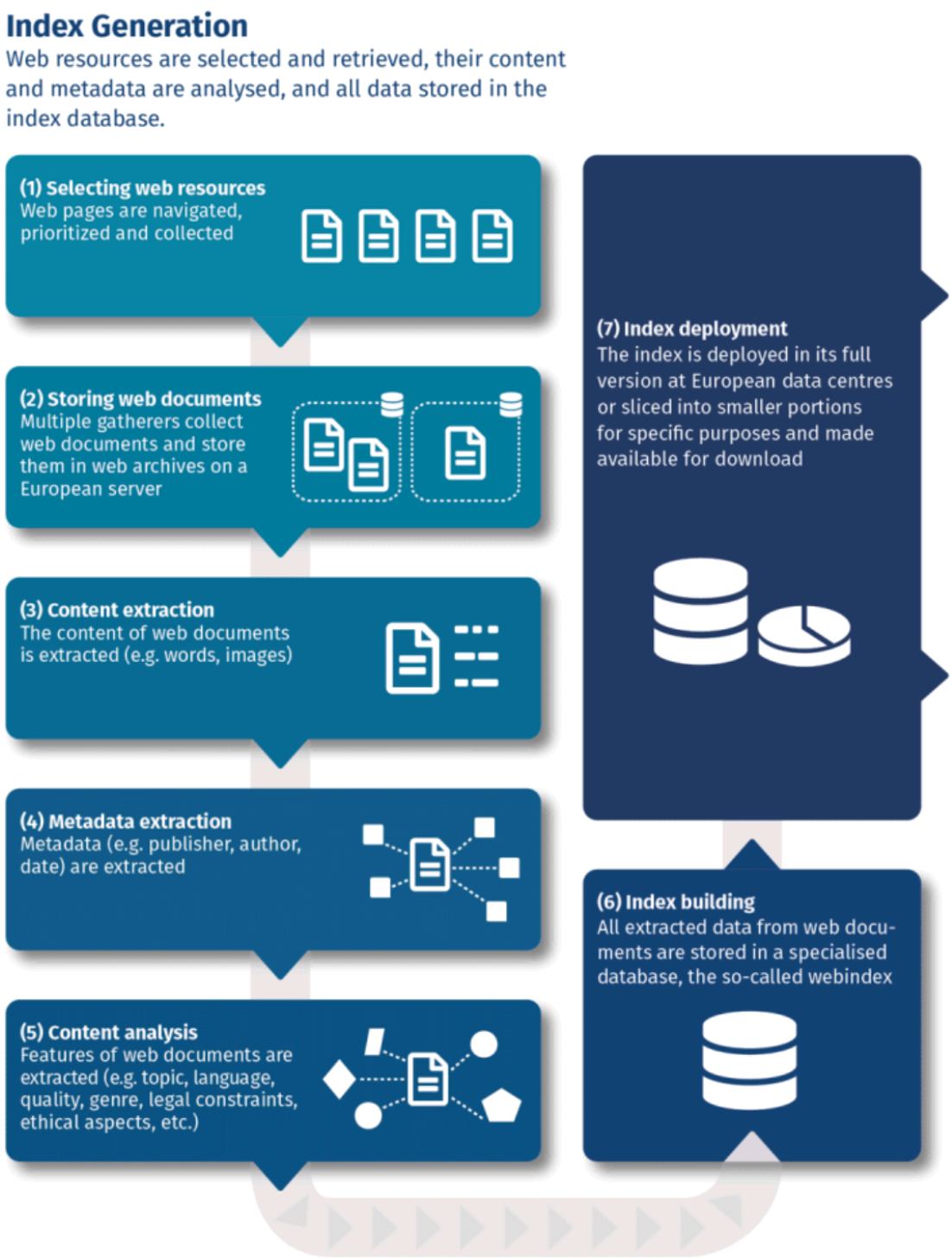
Em nenhuma circunstância se ele deve se tornar uma repetição do conceito do Google ou torná-lo um Google europeu, de acordo com Michael Granitzer, professor da Data Science University de Passau, que está coordenando o projeto OpenWebSearch.

LISTA DE PARCEIROS DO PROJETO

1. Universidade de Passau, Alemanha (uni-passau.de)
2. Centro de Supercomputação Leibniz da Academia de Ciências e Humanidades da Baviera, Alemanha (lrz.de)
3. Fundação da Universidade de Radboud, Holanda (ru.nl)
4. Universidade de Leipzig, Alemanha (uni-leipzig.de)
5. Universidade de Tecnologia de Graz, Áustria (tugraz.at)
6. Centro Aeroespacial Alemão, Alemanha (dlr.de)
7. VSB - Universidade Técnica de Ostrava, IT4Innovations, República Tcheca (www.vsb.cz)
8. Organização Europeia para Pesquisa Nuclear - CERN, Suíça (home.cern)
9. Fundação Open Search, Alemanha (opensearchfoundation.org)
10. A1 Slovenija, telekomunikacijske storitve, d. d., Slovenia (a1.si)
11. CSC-Tieteen Tietotekniikan Keskus Oy, Finlândia (csc.fi)
12. Stichting Nlnet, Holanda (nlnet.nl)
13. Universidade Bauhaus de Weimar, Alemanha (uni-weimar.de)
14. SUMA-EV - Associação para Acesso Livre ao Conhecimento, Alemanha (suma-ev.de)

O projeto envolve a configuração de uma infraestrutura com a qual os mecanismos de pesquisa e outros serviços possam trabalhar.

Uma configuração como a do Google certamente não é a intenção.
Em vez disso, ela terá que crescer como, por exemplo, Wikipedia, que continha um pequeno núcleo e depois aumentou rapidamente de tamanho.
Atualmente, um total de 14 parceiros do projeto está desenvolvendo técnicas de rastreamento.


Index Generation
Web resources are selected and retrieved, their content and metadata are analysed, and all data stored in the index database.
(1) Selecting web resources
Web pages are navigated, prioritized and collected
(2) Storing web documents
Multiple gatherers collect web documents and store them in web archives on a European server
(3) Content extraction
The content of web documents is extracted (e.g. words, images)
(4) Metadata extraction
Metadata (e.g. publisher, author, date) are extracted
(5) Content analysis
Features of web documents are extracted (e.g. topic, language, quality, genre, legal constraints, ethical aspects, etc.)
(6) Index building
All extracted data from web documents are stored in a specialised database, the so-called webindex
(7) Index deployment
The index is deployed in its full version at European data centres or sliced into smaller portions for specific purposes and made available for download

Open WebSearch .eu

Search Applications

A user search request will be answered by a search application that makes use of the open web index.

(a) Selecting web documents
Web documents are selected that fit to the user search request

(b) Ranking web documents
The selected Web documents are sorted (ranked) according to their assumed relevance for the user

(c) Purpose-specific search
An application with user interface enables the search for general or specific purposes

(d) User searches and receives result
The user is supported to better understand the search process

Data Products

Knowledge representation models will be created using the open web index, in order to be used by any agent and for many applications

Building knowledge graphs
Using the extracted information from web documents, a knowledge graph is created that supports specific search requests

Building AI Language Models
Creation of different types of language models by using Web documents

Any agent, multiple applications
Language models and knowledge graphs can be used by any agent (or application)

Tecnicamente:
O rastreamento se concentra na coleta de conteúdo de uma página da Web, enquanto a indexação tem uma função diferente.
A indexação se concentra no armazenamento e na organização do conteúdo.
Durante esse processo, sua relevância para determinadas pesquisas é determinada.
Esses parceiros - onde você pode encontrar os logotipos dos endereços neste artigo - selecionam metadata* que o índice deve conter e, assim, projetar uma distribuição descentralizada do OWI em diferentes servidores e locais na Europa. *(Metadata são definidos como os dados que fornecem informações sobre um ou mais aspectos dos dados; são usados para resumir informações básicas sobre os dados que podem facilitar o rastreamento e o trabalho com dados específicos. Alguns exemplos são: Modo de criação dos dados. Finalidade dos dados. Hora e data de criação.*)

PARTICIPANTES (PARCEIROS):
Os parceiros de infraestrutura participantes incluem
1. Centro de Supercomputação Leibniz, em Munique,
2. CSC em Espoo, Finlândia, que opera o maior supercomputador da Europa,
3. Centro Nacional de Supercomputação da República Tcheca IT4Innovations e
4. CERN perto de Genebra.
5. Outros parceiros incluem aFundação Holandesa da Universidade Radboud e Stichting Nlnet.

DATA COLLECTION

Ao direcionar os dados orquestrados por meio dessa página de índice, você mesmo pode inovar nessa área, por exemplo, diversificando o uso do mecanismo - como todos os problemas devem ser resolvidos - tornando-os menores:
usando várias categorias de mecanismos de pesquisa que são muito melhores e mais rápidos em fornecer respostas incisivas e até mesmo previsões não solicitadas em suas próprias áreas.
Se for constatado que há um grande número de infecções por gripe aviária na web, se eles também se tornarem transmissíveis (preferencialmente coletados por meio de conhecimento científico) então, como sociedade, você pode responder muito rapidamente.
Parece óbvio que a IA pode estar envolvida nisso, mas isso deve ser feito sob controle rigoroso, e a Europa está pronta para isso.

Além disso, é possível desenvolver novas facetas de linguagem usando um índice extenso com base principalmente em fontes europeias. Em particular, idiomas menores ou combinações de idiomas, como Lithuanian, Basque e Frisian, que são rapidamente ignorados por serviços orientados globalmente, podem se beneficiar.

Muitas novas possibilidades estão surgindo:
fazer um balanço do fluxo real de pessoas fora da Europa e dar a ele um estágio por meio disso, permitindo um melhor controle da disseminação e, acima de tudo, verificando de quais indivíduos precisamos, para que tenhamos uma descoberta da verdade muito mais clara, todos porque esse é um índice relacionado à Europa. Vejo um número infinito de benefícios.

RASTREADORES EDUCADOS:

Espera-se que os rastreadores da Web usados sejam "organizados e, acima de tudo, respeitosos". Isso tem um bom motivo.
De acordo com Stefan Voigt, o presidente da Open Search Foundation que introduziu o projeto OpenWebSearch, o rastreamento da web é responsável por cerca de 30 a 40 por cento da carga total da rede, se o streaming* não estiver incluído.

(**Mídia de streaming é multimídia que pode ser reproduzida com um reprodutor de mídia off-line ou on-line. Tecnicamente, o fluxo é fornecido e consumido de forma contínua por um cliente, com pouco ou nenhum armazenamento intermediário nos elementos da rede. O streaming refere-se ao método de entrega do conteúdo, e não ao conteúdo em si.* **Consome muita energia**).

Esse é um custo significativo para os hosts da Web.
Além disso, um site fica facilmente sobrecarregado se todas as páginas HTML forem recuperadas em paralelo (simultaneamente) com vários servidores.
Devido a um erro de implementação, isso aconteceu quando os pesquisadores do projeto rastrearam as páginas do Bank of America. O banco entendeu isso erroneamente como um ataque de negação de serviço e bloqueou os servidores envolvidos.

Com os chamados Crawling Politeness Rules, esses mal-entendidos devem, em princípio, ser evitados. Por exemplo, nunca mais do que uma solicitação por segundo deve ser iniciado e as restrições de página do arquivo robots.txt do operador devem, obviamente, ser observadas. Isso permite que os operadores de sites perguntem "bots" para não visitar determinadas partes do site.
A análise dos rastreamentos já revela outra desvantagem em comparação com o índice dominado pelo Google:
A edição e o particionamento robustos de arquivos HTML é um problema que não pode ser simplesmente resolvido tecnicamente, mas somente em colaboração com os operadores do site. Para isso, o Google oferece a eles o Google Search Console, uma ferramenta de análise que permite que eles realizem a otimização de mecanismos de pesquisa (SEO) por conta própria e, assim, adaptar suas páginas especificamente ao bot do Google.

Lá, os webmasters podem, por exemplo, ver rapidamente se o texto principal de uma página da Web está onde o Google espera que ele esteja.

A análise produz uma árvore de análise completa?
Quais palavras-chave o Reconhecimento de bots do Google em páginas da web?

A maioria dos proprietários de sites tem páginas que contêm opiniões e podem levar a uma linguagem tendenciosa:
(*Viés é um peso desproporcional a favor ou contra uma ideia ou coisa, geralmente de forma imprecisa, fechada, tendenciosa ou inadequada*).
Com o tempo, um público, Índice da web construído de forma transparente poderia mudar completamente o cenário de SEO.

Hoje em dia, quase todo empresário está tentando otimizar seu site para obter uma classificação elevada nos resultados de pesquisa do Google.

Se muitos serviços de pesquisa especializados substituíssem esse monopólio, a situação mudaria radicalmente. O melhor seria que os proprietários de sites e os usuários da Web se concentrassem na criação do conteúdo de seus sites.

Quando a pesquisa na Web não for mais dominada por um monopolista, tudo o que importa é preparar o conteúdo da maneira mais estruturada possível e apoiá-lo com metadados significativos.

NÃO HÁ ANÁLISE DE USUÁRIO
A seleção dos fatores de pesquisa a serem incluídos no novo índice da Web ainda está em aberto e sob intensa discussão.
Além dos direitos de uso do conteúdo, os parceiros discutem principalmente fatores de conteúdo, como qualidade do conteúdo, gêneros e uma ponderação de classificação de página de estruturas de links (URLs).
Há também fatores técnicos, como tempos de resposta.
Projetos de pesquisa empolgantes dizem respeito à ideia de georreferenciamento, (indicando a localização por meio de coordenadas, por exemplo) que permite serviços da Web para cidades ou comunidades individuais.
No entanto, a classificação espacial de links da Web exige primeiro uma análise adequada, pois os Estados Unidos não estão apenas nos Países Baixos.

UMA DIFERENÇA MUITO IMPORTANTE EM RELAÇÃO AOS GRANDES ÍNDICES ATUAIS:

cliques de usuários e outras análises de usuários não chegam à OWI.
Isso dificilmente é possível, pois os parceiros do projeto não pretendem operar mecanismos de busca e, portanto, coletar dados de usuários.
'Os serviços da Web que se baseiam na OWI podem, é claro, sempre registrar o termo de pesquisa e o comportamento do usuário em sua plataforma,' explica o consultor técnico da organização sem fins lucrativos Suma, que opera o mecanismo de metabusca MetaGer.

**(O MetaGer é um mecanismo de metabusca voltado para a proteção da privacidade dos usuários. Com sede na Alemanha e hospedado como uma cooperação entre a ONG alemã "SUMA-EV - Association for Free Access to Knowledge" e a Universidade de Hannover, o sistema é construído com base em 24 rastreadores da Web de pequena escala sob o controle do próprio MetaGer. Em setembro de 2013, a MetaGer lançou o MetaGer.net, uma versão em inglês de seu mecanismo de busca*).

Até certo ponto, isso é até necessário, por exemplo, para poder bloquear pesquisas em massa de spammers ou para analisar e aprimorar seu próprio serviço.
Mas, sempre, os usuários terão a liberdade de decidir a quem confiarão seus dados de uso.

DISTRIBUÍDO DE FORMA SEGURA

Por causa da infraestrutura crítica Para a Europa, planeja-se distribuir o OWI em vários países. Dessa forma, assim como as bibliotecas nacionais, a OWI poderia ser armazenada em diferentes centros de dados. Os índices podem ser desenvolvidos em diferentes países com um foco regional, por exemplo, por idioma de origem.

Além disso, diferentes organizações poderiam assumir a manutenção de diferentes subíndices e hospedá-los, por exemplo, um índice para geociências ou um índice para o setor financeiro mercados, para distribuição verde e plantio e fluxo de água. Um dos problemas na maioria dos países é uma visão geral ruim ou ausente de todos os
dados do duto que se encontram tridimensionalmente no solo (seria um grande avanço para o planejamento urbano, pois os danos financeiros e sociais podem ser previstos e evitados).

O CERN, por exemplo, já manifestou interesse em compilar e gerenciar todas as informações sobre física de partículas. O projeto não ficará apenas nas mãos da ciência.
Há fundos para o projeto e as pessoas precisam de agentes econômicos.
Não é necessário apenas ter uma mente de pesquisador; também é preciso ser empreendedor e ter ideias de negócios para criar, por exemplo, novos mecanismos de pesquisa e serviços baseados na OWI.
Porque o crescimento a médio prazo do índice da Web só é possível se a receita de licenciamento pagar os custos de infraestrutura.

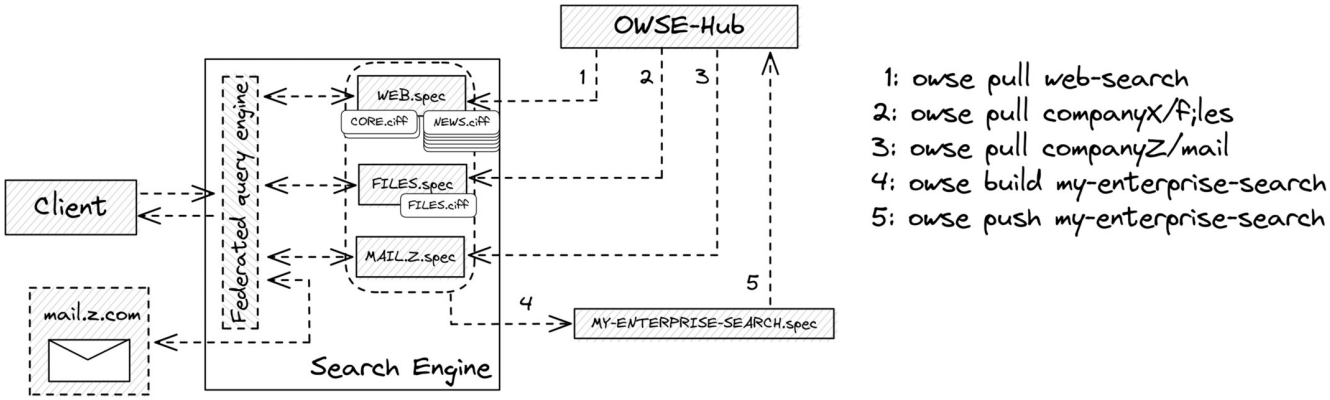
POOL PARA MODELOS DE LINGUAGEM (LLCs e idiomas reais)

Logo no início do projeto e, portanto, antes mesmo do hype em torno do ChatGPT, os parceiros consideraram a Índice da Web aberta, com seu foco em conteúdo e idiomas europeus, para ser um pool de dados para modelos de idiomas especializados.
Novos mecanismos de pesquisa também poderiam usar imediatamente esses modelos como interface para pesquisas.

Os usuários geralmente não estão procurando links, mas por respostas às suas perguntas ou mesmo sugestões de soluções.

Isso fala a favor do uso de chatbots.

O mecanismo de busca Bing, com seus ChatGPT interface, e Google, com seus Gêmeos, parecem estar prontos para superar essa abordagem no lado direito.
Coordenador do projeto Granitzer também vê o mecanismo de busca You.com como um exemplo bem-sucedido de como um chatbot da IA pode capturar consultas de pesquisa e descrever links de resultados.
No entanto, não se pode confiar nas respostas de AI no momento.
Além disso, ainda não foram realizados estudos sobre o custo e a escalabilidade de tais sistemas.
Os modelos de linguagem como interface terão de ser integrados aos mecanismos de busca no futuro. Porque essa tecnologia oferece uma maneira de processar consultas de pesquisa em linguagem natural ou como um chatbot e resumir os resultados.
Portanto, eles são a melhor interface para iniciar uma visão geral.



NOVOS SERVIÇOS NA INTERNET

O OpenWebSearch O projeto está buscando novos parceiros para agregar seus conhecimentos. Já existem discussões com serviços especializados, como um léxico de vocabulário, que reconhece novas criações de palavras, metáforas e expressões idiomáticas automaticamente e ao vivo usando um índice atualizado e de livre acesso na Web.
Outro projeto, o Europe Media Monitor, examina tópicos de tendências, eventos atuais e desenvolvimentos na Internet.
https://knowledge4policy.ec.europa.eu/online-resource/europe-media-monitor-emm_en
Novos serviços poderiam, por exemplo, listar os prós e os contras de tópicos controversos em vez de mostrar resultados de pesquisa, como fazem o Google e seus concorrentes.
Esse já é o caso no Args.me website. (muito difícil de alcançar!)
Além disso, o OWI poderia formar a base para serviços de pesquisa específicos de um determinado tópico.
Até mesmo a criação de mecanismos de pesquisa dedicados para telefones celulares pode ser o resultado da limitação do índice de pesquisa.
Como resultado, os usuários podiam pesquisar localmente sem acesso à Internet e só precisariam revelar seus dados de usuário no dispositivo móvel.
Você pode se imaginar como um hub de mecanismo de pesquisa próximo à OWI um dia.
O usuário escolhe suas áreas de interesse e pode usar vários recursos, como a pesquisa de texto completo. Ele limita os resultados a fontes que são particularmente confiáveis ou populares.
No final das contas, ele recebe um mecanismo de busca configurado de acordo com suas preferências depois que ele enviar seu formulário.

INFRAESTRUTURAS ESSENCIAIS

O projeto OpenWebSearch da UE durará inicialmente até setembro de 2025.
Nesse período, os parceiros pretendem criar o índice básico de cinco petabytes. Depois disso, o foco será o financiamento de infraestrutura sustentável, provavelmente por meio de financiamento adicional da UE.

CONCLUSÃO

O Open Web Index é uma infraestrutura essencial para a soberania digital da Europa.
Os parceiros do projeto esperam que ele crie estruturas transparentes na Internet.
O **Índice da Web Europeu** previsto tem o objetivo de criar mais diversidade e provavelmente será particularmente útil para aqueles que desejam oferecer apenas as melhores e mais confiáveis informações em seus sites.

O conteúdo deste artigo foi composto por artigos originários destes endereços:

https://asistdl.onlinelibrary.wiley.com/doi/10.1002/asi.24818 /
https://openwebsearch.eu / https://openwebsearch.eu/community/3rdparty-calls/call3/
https://open-console.eu/ https://zenodo.org/communities/owseu/records?q=&l=list&p=1&s=10&sort=newest /

https://openwebsearch.eu/partners/ bem como outros artigos não mencionados aqui.

Faça uma doação para a Ucrânia e obtenha uma licença gratuita em:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

Se o senhor for de origem ucraniana, poderá obter uma assinatura gratuita da Blaise Pascal Magazine, e também lhe daremos uma versão em pdf gratuita do Lazarus Handbook. O senhor precisa nos enviar seu nome ucraniano e seu endereço de e-mail ucraniano (que ainda funcione para o senhor), para que seja comprovado que o senhor é ucraniano de verdade. Envie-os para editor@blaisepascal.eu e o senhor receberá o livro e a assinatura.Se o senhor for de origem ucraniana, poderá obter uma assinatura gratuita da Blaise Pascal Magazine, e também lhe daremos uma versão em pdf gratuita do Lazarus Handbook. O senhor precisa nos enviar seu nome ucraniano e seu endereço de e-mail ucraniano (que ainda funcione para o senhor), para que seja comprovado que o senhor é ucraniano de verdade. Envie-os para editor@blaisepascal.eu e o senhor receberá o livro e a assinatura.

BLAISE PASCAL MAGAZINE

Multi platform /Object Pascal / Internet / JavaScript / Web Assembly / Pas2Js / Databases / CSS Styles / Progressive Web Apps Android / IOS / Mac / Windows & Linux

Blaise Pascal

COMPONENTS 4 DEVELOPERS

Faça uma doação para a Ucrânia e obtenha uma licença gratuita em:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

COMPONENTS 4 DEVELOPERS

D11

Faça uma doação para a Ucrânia e obtenha uma licença gratuita em:

https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/

kbmMW Professional e Enterprise NOVO EDITION V. 5.23 kbmMemTable NOVO EDITION V. 7.99.00 Standard e Professional Edition

O 5.23.00 é uma versão que contém novidades, refinamentos e correções de bugs, além do suporte ao SSL v 3, suporte ao WebSoft, outras melhorias no SmartBind, novos algoritmos de hashing de alta performance, melhorias no RemoteDesk para amostragem e muito mais. Essa versão exige o uso do kbmMemTable v. 7. 9 8 .0 0 ou mais recente.

- Suporte ao RAD Alexandria
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 e
- Suporte a cliente e servidor OS X
- Servidor de aplicativos nativo de alto desempenho 100% definido pelo desenvolvedor servidor de aplicativos definido pelo desenvolvedor
- Suporte completo para balanceamento de carga centralizado e distribuído balanceamento de carga centralizado e distribuído e fail-over
- Suporte avançado a ORM/OPF, incluindo suporte a bancos de dados existentes
- Suporte avançado a registro em log
- Estrutura de configuração avançada
- Suporte avançado a agendamento para acesso fácil a programação de vários threads
- Serviço inteligente avançado e clientes para facilitar a publicação muito fácil da funcionalidade
- Funções aleatórias de alta qualidade.
- Geradores de senhas pronunciáveis de alta qualidade.
- Compactação de LZ4 e Jpeg de alto desempenho
- Estrutura completa de notação de objetos, incluindo suporte total a suporte total a YAML, BSON, Messagepack, JSON e XML
- Marshalling avançado de objetos e valores de e para
- YAML, BSON, Messagepack, JSON e XML
- Suporte a transporte TCP nativo de alto desempenho
- Transporte HTTPSys de alto desempenho para Windows.
- Suporte a CORS em serviços REST/HTML.
- Suporte nativo a clientes PHP, Java, OCX, ANSI C, C# e Apache Flex!

O kbmMemTable é a tabela de memória mais rápida e com mais recurso para produtos Embarcadero.

- Suporta facilmente grandes conjuntos de dados com milhões de registros
- Suporte fácil a streaming de dados
- Opção de usar o mecanismo SQL nativo
- Suporte a transações aninhadas e desfazer
- Construção nativa e rápida em M/D, agregação/agrupamento
- Recursos de seleção de intervalo
- Recursos avançados de indexação para desempenho extremo

- Novo: suporte total a Web-socket.
  A próxima versão do kbmMW Enterprise Edition incluirá várias novidades e aprimoramentos.
  Um deles é o suporte total a Web-sockets.
- Nova estrutura de internacionalização sensível ao contexto I18N para tornar seus aplicativos multilíngues.
- Novo suporte ORM LINQ para exclusão e atualização.
- Suporte a comentários em YAML.
- Novo suporte a StreamSec TLS v4 (por StreamSec).
  Muitos outros aprimoramentos e correções de recursos.

Acesse http://www.components4developers.com
Para obter mais informações sobre a kbmMW

- Acesso a banco de dados unificado e de alta velocidade (mais de 35 APIs de banco de APIs de banco de dados compatíveis) com pooling de conexões, metadados e cache de dados e metadados em todas as camadas
- Acesso de vários cabeçalhos ao servidor de aplicativos, via REST/AJAX, binário nativo, Publish/Subscribe, SOAP, XML, RTMP a partir de navegadores da Web, dispositivos incorporados, servidores de aplicativos vinculados, PCs, dispositivos móveis, sistemas Java e muitos outros clientes
- Suporte completo para hospedagem de aplicativos baseados em FastCGI (normalmente PHP/Ruby/Perl/Python)
- Suporte nativo completo ao AMQP 0.91 (Advanced Message Queuing Protocol)
- Área de trabalho remota completa, segura e com marca, de ponta a ponta, com vídeo HD quase em tempo real, suporte a 8 monitores, detecção de textura, compactação e compartilhamento de área de transferência.
- Pacote do kbmMemTable Professional, que é a tabela de memória mais rápida e mais rica em recursos em tabela de memória para produtos Embarcadero.

COMPONENTS 4 DEVELOPERS