

Guia prático para criação de um MVP de um serviço de notícias  
baseado em web e aplicativos combinando-se conceitos,  
tecnologias e ferramentas simples



Maio de 2018

Bruno Pinto Ferraz Fabbri

## Sumário

<b>Guia prático para criação de um MVP de um serviço de notícias baseado em web e aplicativos combinando-se conceitos, tecnologias e ferramentas simples.....</b>	<b>1</b>
<b>1. Objetivos do guia .....</b>	<b>4</b>
1.1. O que faz parte dos objetivos do guia: .....	4
1.2. O que não faz parte dos objetivos do guia: .....	4
<b>2. Objeto de estudo .....</b>	<b>5</b>
<b>3. Tecnologias, linguagens, produtos e serviços utilizados .....</b>	<b>6</b>
3.1. Google Domains.....	6
3.2. DreamHost.....	6
3.3. FileZilla.....	6
3.4. MySQL Workbench.....	6
3.5. Sublime Text .....	7
3.6. XAMPP .....	7
3.7. Apache Server .....	7
3.8. MariaDB.....	7
3.9. PHP .....	7
3.10. HTML .....	7
3.11. CSS.....	8
3.12. Javascript .....	8
3.13. jQuery.....	8
3.14. Bootstrap .....	8
3.15. Adobe PhoneGap .....	9
3.16. Adobe PhoneGap Build.....	9
3.17. Google Chrome .....	9
<b>4. Pontos relevantes sobre o desenvolvimento do MVP.....</b>	<b>11</b>
<b>5. Desenvolvendo o MVP na prática .....</b>	<b>13</b>
5.1. Setup inicial do ambiente de desenvolvimento local .....	13
5.2. Contratação e configuração do servidor .....	13
5.3. Comprando um domínio e configurando o DNS .....	23
5.4. Criando as conexões com o banco de dados local e de produção .....	26
5.5. Criando um novo <i>schema</i> para o banco de dados .....	29
5.6. Sincronizando o novo <i>schema</i> com o banco de dados .....	31
5.7. Código, parte 0: Entendendo a arquitetura.....	33
5.8. Código, parte 1: <i>Backend</i> em PHP .....	33
5.9. Código, parte 2: painel administrativo “em web” .....	42
5.10. Código, parte 3: app “em web” .....	52
5.11. Colocando em produção, parte 1: servidor ( <i>backend</i> ) e interface administrativa.....	63
5.12. Colocando em produção, parte 2: app .....	65

## **Lista de Figuras**

Figura 1 - Visão geral da arquitetura do MVP .....	5
Figura 2 - Dashboard Dreamhost: Manage Domains .....	14
Figura 3 – Add hosting to a Domain / Sub-Domain .....	15
Figura 4 – Configurando um novo domínio, parte 1 .....	16
Figura 5 – Configurando um novo domínio, parte 2 .....	17
Figura 6 – Sucesso ao configurar um novo domínio.....	18
Figura 7 – Dashboard Dreamhost: MySQL Databases.....	19
Figura 8 – Add New Hostname .....	19
Figura 9 – Criando um novo hostname .....	20
Figura 10 – Criando uma nova base de dados.....	21
Figura 11 – Sucesso ao criar uma nova base de dados .....	21
Figura 12 – Permissões de acesso para o usuário, parte 1.....	22
Figura 13 – Permissões de acesso para o usuário, parte 2.....	23
Figura 14 – Configurando o DNS, parte 1 .....	24
Figura 15 – Configurando o DNS, parte 2 .....	25
Figura 16 – DNS configurado com sucesso.....	26
Figura 17 – Criando uma nova conexão com o banco de dados, parte 1.....	26
Figura 18 – XAMPP: MySQL Database + Apache Web Server .....	27
Figura 19 – Criando uma nova conexão com o banco de dados, parte 2.....	28
Figura 20 – Criando uma nova conexão com o banco de dados, parte 3.....	28
Figura 21 – Criando um novo schema, parte 1.....	29
Figura 22 – Criando um novo schema, parte 2.....	29
Figura 23 – Criando um novo schema, parte 3.....	29
Figura 24 – Criando um novo schema, parte 4.....	30
Figura 25 – Criando um novo schema, parte 5.....	30
Figura 26 – Sincronizando o schema com o banco.....	31
Figura 27 – Revisando o script de sincronização .....	32
Figura 28 – Schema sincronizado com sucesso .....	32
Figura 29 – Hierarquia de pastas e arquivos do backend PHP .....	34
Figura 30 – Hierarquia de pastas e arquivos da interface administrativa .....	43
Figura 31 – Interface administrativa rodando em ambiente local .....	51
Figura 32 – Painel de desenvolvedor do Google Chrome .....	52
Figura 33 – Estrutura de pastas e arquivos do app .....	53
Figura 34 – App sendo testado no navegador.....	61
Figura 35 – Navegador simulando um device (iPhone 4).....	62
Figura 36 – Alterando as permissões da pasta uploads .....	64
Figura 37 – Painel administrativo em produção.....	65
Figura 38 – Aplicativo Android (.apk) criado no PhoneGap Build .....	66
Figura 39 - Instalando app no Android - passo 1.....	67
Figura 40 - Instalando app no Android - passo 2 .....	67
Figura 41 - Instalando app no Android - passo 3 .....	68
Figura 42 - Instalando app no Android - passo 4 .....	68
Figura 43 - Instalando app no Android - passo 5 .....	69
Figura 44 - Instalando app no Android - passo 6 .....	69
Figura 45 - Instalando app no Android - passo 7 .....	70
Figura 46 - Instalando app no Android - passo 8 .....	70
Figura 47 - Instalando app no Android - passo 9 .....	71

## 1. Objetivos do guia

### 1.1. O que faz parte dos objetivos do guia:

- Demonstrar como integrar diversos conceitos, ferramentas e tecnologias simples e pontuais para se criar um MVP completamente funcional, através de um exemplo;
- Explicar brevemente o funcionamento de cada uma dessas partes;
- Mostrar que um MVP pode ser criado rapidamente e sem a necessidade de aprender a lidar com diversos frameworks e demais modismos do mundo da tecnologia;
- Ir além do código e mostrar o passo a passo também da parte burocrática e de setup.

### 1.2. O que não faz parte dos objetivos do guia:

- Ensinar lógica de programação, conceitos aprofundados sobre a internet ou linguagens de programação em geral. Assume-se que o estudante já teve contato com esses temas em alguma disciplina ou em algum curso disponível na web;
- Entrar em detalhes a respeito do código desenvolvido ao longo do guia ou das ferramentas e serviços utilizados;
- Criar um MVP “perfeito”, seja por conta de tratamento de erros no código, escolha das linguagens, ferramentas, ambiente de desenvolvimento, servidor, paradigma, entre outros;
- Elencar exaustivamente ou comparar as opções existentes para fazer o papel de cada uma das partes escolhidas para integrar este MVP.

## 2. Objeto de estudo

Para servir de exemplo de MVP, foi eleito um serviço de veiculação de notícias de um jornal local, no qual há um aplicativo para o usuário consultar as notícias e uma interface administrativa para publicação das notícias por parte do jornal. O aplicativo funciona tanto em celulares quanto em tablets iOS e Android e a interface administrativa é web. Uma visão geral da arquitetura do MVP é a seguinte:

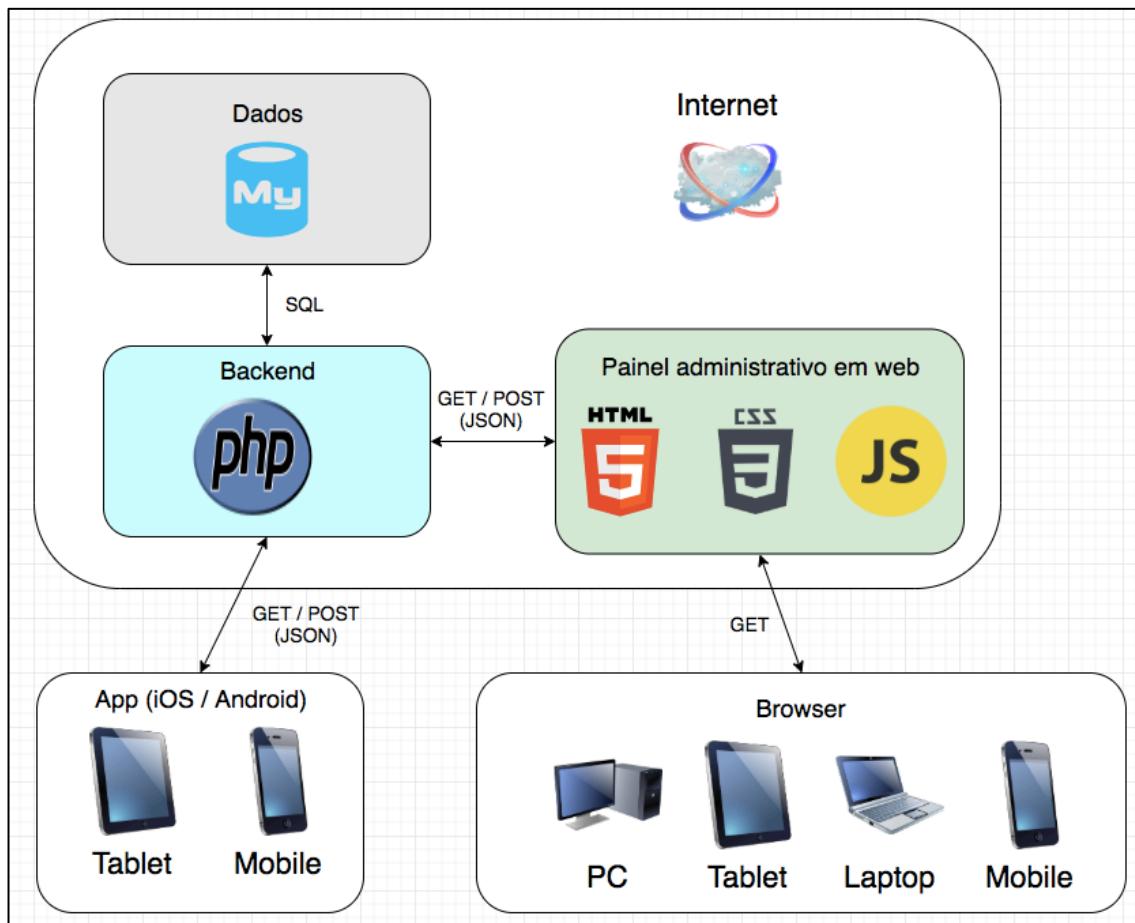


Figura 1 - Visão geral da arquitetura do MVP

### 3. Tecnologias, linguagens, produtos e serviços utilizados

Nesta seção são elencadas as tecnologias, linguagens, produtos e serviços utilizados ao longo do guia. É válido destacar que com exceção de poucos serviços, que são pagos, todas as tecnologias utilizadas são *open-source* e seus mantenedores merecem um agradecimento especial, pois sem elas não seria possível desenvolver este projeto (e a maior parte da web) de forma tão rápida e simples. Ainda nesse sentido, fica registrada aqui uma sugestão prática e efetiva de agradecimento: comprar licenças das ferramentas que lhe forem úteis, mesmo que elas possuam uma versão de teste gratuita. Sem mais delongas, a lista:

#### 3.1. Google Domains

<https://domains.google.com/> – Serviço para registro de domínios. Serve para registrar a maior parte dos tipos de domínio de nível superior (.com, .org, .site, etc.). Esses serviços são boas fontes de pesquisa e até de inspiração para nomes de negócios. Vale lembrar que os preços não são fixos, então após escolhido o domínio, pesquise em outros serviços concorrentes.

#### 3.2. DreamHost

<https://www.dreamhost.com/> – Serviço de hospedagem web simples para PHP, com bom atendimento e relativamente barato. Existem vários outros disponíveis, inclusive serviços gratuitos ou com *free-tier* para projetos pequenos ou tempo determinado, por exemplo: Heroku ou Amazon (Foi escolhido o Dreamhost, pois ele é simples e o autor já possui uma conta e créditos nele).

#### 3.3. FileZilla

<https://filezilla-project.org/> – Solução gratuita de FTP simples, leve e que roda em qualquer plataforma. Licença: GNU GPL. Versão utilizada: 3.32.0.

#### 3.4. MySQL Workbench

<https://dev.mysql.com/downloads/workbench/> – O MySQL Workbench possui um ambiente com ferramentas para modelagem, desenvolvimento, administração e migração de bases de dados. Basicamente, ele é uma solução mais robusta e mais segura que o tradicional phpMyAdmin. Licença: GPL. Versão utilizada: 6.3.10 build 12092614 CE (64 bits) Community.

### 3.5. Sublime Text

<https://www.sublimetext.com/> – Um editor de texto leve, mas ao mesmo tempo sofisticado que é bom tanto para linguagens de programação quanto de marcação. Ele possui *features* providenciais e a possibilidade de instalar *snippets* e *plugins* através de um gerenciador de pacotes interno. Licença: Ele pode ser baixado para teste, mas precisa ser comprado para uso contínuo, sendo que, atualmente, não existe um tempo máximo pré-determinado para esse teste. Versão utilizada: 3.0 build 3143.

### 3.6. XAMPP

[https://www.apachefriends.org/pt\\_br/index.html](https://www.apachefriends.org/pt_br/index.html) – O XAMPP é o ambiente mais popular para desenvolvimento PHP. Ele é gratuito e fácil de instalar em qualquer plataforma. Sua sigla representa Apache + MariaDB + PHP + Perl (que não será usado neste guia). Licença: GNU. Versão utilizada: 5.6.24-1.

### 3.7. Apache Server

<https://httpd.apache.org/> – O servidor HTTP mais popular da web, um projeto *open-source* mantido pela [The Apache Foundation](#). Licença: Apache 2.0. (Não será necessário instala-lo separadamente, pois ele vem no XAMPP)

### 3.8. MariaDB

<https://mariadb.org/> – O MariaDB é um *fork* do MySQL criado pelos próprios desenvolvedores do MySQL após este ser comprado pela Oracle. Ele é *open-source* e um dos SGBDs mais populares. Licença: GNU 2. (Não será necessário instala-lo separadamente, pois ele vem no XAMPP)

### 3.9. PHP

<https://secure.php.net/> – Uma linguagem interpretada de propósito geral muito popular que é especialmente adequada ao desenvolvimento web. Licença: PHP v3.01. (Não será necessário instala-lo separadamente, pois ele vem no XAMPP)

### 3.10. HTML

<https://www.w3.org/standards/webdesign/htmlcss> – Acrônimo para HyperText Markup Language, o HTML é a principal linguagem para criação e organização de conteúdo na web. Ao contrário do que muitos pensam, ele não é uma linguagem de programação, mas sim uma linguagem de marcação que

serves para prover a estrutura de uma página. Ele também não é “instalável”, mas sim interpretada pelos *browsers*. Os padrões do HTML são mantidos pelo W3C (*World Wide Web Consortium*) e sua licença depende de quem o está implementando.

### 3.11. CSS

<https://www.w3.org/standards/webdesign/htmlcss> – Sigla para *Cascading Style Sheets*, o CSS, juntamente com o HTML, é uma das principais tecnologias para a construção de páginas web. Ele é responsável por descrever como a estrutura da página vai ser estilizada visualmente, como será seu layout e definir alguns comportamentos. Os padrões do CSS são mantidos pelo W3C (*World Wide Web Consortium*) e sua licença depende de quem o está implementando.

### 3.12. Javascript

Uma linguagem de programação que já vem embarcada em nos *browsers*. Suas bases são mantidas e padronizadas pela Ecma International (<https://www.ecma-international.org/>) e sua licença depende de quem o está implementando. A maior parte do código apresentado neste tutorial está em “javascript puro”, ou Vanilla JS. Ele é a base para a implementação de frameworks e bibliotecas javascript, como o jQuery, que também é usado aqui (O site <http://vanilla-js.com/> até trata isso de forma cômica). Versão utilizada: ECMAScript 5.

### 3.13. jQuery

<http://jquery.com/> – O jQuery é uma biblioteca javascript que aumenta o nível de abstração para quem vai escrever código javascript. Ele facilita a escrita de alguns padrões, encurta código, compatibiliza o uso entre diversos navegadores e fornece algumas operações comumente necessárias. Porém, ele acaba deixando o código ligeiramente mais lento em alguns casos, pois insere mais uma camada de abstração que, no fim das contas, utilizará javascript puro para realizar suas operações. Licença: MIT e GPL. Versão utilizada: 3.3.1.

### 3.14. Bootstrap

<https://getbootstrap.com/> – O Bootstrap é a biblioteca *open-source* de componentes *front-end* para web mais popular do mundo. Ele ajuda a tornar seu projeto responsivo e *mobile-first* fácil e rapidamente. Licença MIT. Versão

utilizada: 4.1.1. Para que todas as funcionalidades da v4.1.1 funcionem, ele necessita também das bibliotecas javascript jQuery e popper. As versões utilizadas delas foram 3.3.1 e 1.14.3, respectivamente.

### 3.15. Adobe PhoneGap

<https://phonegap.com/> – O framework Adobe PhoneGap é uma distribuição *open-source* do Cordova, trazendo o benefício da junção de forças de um time de profissionais e de uma comunidade forte de desenvolvedores. Ele utiliza código web (HTML + CSS + JS) para gerar aplicativos para diversas plataformas, incluindo iOS e Android. Com isso, você dispensa aprender (ou contratar quem saiba) Objective-C ou Swift, para programar para iOS, e Java, para programar para Android. Você usa apenas web, que você “já sabe”, e mantém apenas uma base de código para todas as plataformas, tornando o desenvolvimento do MVP mais rápido e mais barato. Licença: Apache 2.0. Versão utilizada: 8.0.0.

### 3.16. Adobe PhoneGap Build

<https://build.phonegap.com/> – O Adobe PhoneGap Build é um serviço em nuvem que cuida da compilação dos aplicativos web, recebendo um pacote .zip e devolvendo os instaláveis para as diversas plataformas. A grande vantagem desse serviço é que assim você não precisa manter versões locais dos diversos SDKs e nem dos demais softwares que seriam necessários para a compilação local dos seus apps com PhoneGap, como Android Studio, para Android, e Xcode, para iOS. O PhoneGap Build é um serviço pago, mas que possui uma *free-tier* que atende nossa necessidade.

### 3.17. Google Chrome

<https://www.google.com.br/chrome/> – O Google Chrome é um dos navegares mais famosos e mais modernos da atualidade. Ele será utilizado para desenvolvimento local, já que usaremos tecnologia web. Suas ferramentas para desenvolvedores são muito boas e a *engine* javascript que ele utiliza é similar às da maioria dos dispositivos Android, o que, de certa forma, garante compatibilidade do nosso código testado no navegador com os aplicativos que rodarão em dispositivos Android. Vale destacar aqui que existem algumas diferenças para as *engines* javascript que rodam no Safari e nos dispositivos iOS, fazendo com que, raramente, haja incompatibilidades. De qualquer forma, é

recomendável compilar apps de teste e testá-los em dispositivos reais sempre que possível. Licença: *freeware* sob os Termos de serviço do Google Chrome. Versão utilizada: 66.0.3359.139, 64 bits.

Por fim, como sabemos que cada software possui apresentações e funcionalidades ligeiramente diferentes para cada plataforma, vale mencionar que o hardware utilizado no desenvolvimento foi um MacBook Air rodando macOS High Sierra v10.13.4.

#### 4. Pontos relevantes sobre o desenvolvimento do MVP

Algumas considerações devem ser feitas nesse momento para que o estudante entenda o contexto desse guia e possa aproveita-lo melhor.

Inicialmente, vale ressaltar que o intuito do guia é ser bastante prático, de forma a possibilitar que o estudante realmente possa aprender todos os passos necessários para o desenvolvimento de um MVP e que ele possa, inclusive, replicar o código aqui apresentado no nosso exemplo. Para tanto, o guia conta com muitos “prints” para facilitar o entendimento dos passos. Consequentemente, com as constantes atualizações dos diversos softwares, os *prints* logo ficarão desatualizados. Ou seja, concentre-se em entender o que acontece por trás de cada parte do “quebra cabeça” para que você possa pesquisar e aprender o que fazer em novas versões, outras plataformas ou softwares substitutos.

Outro ponto importante é que o passo a passo apresentado aqui funciona na ordem em que será apresentado, mas não necessariamente precisa ser seguido apenas nessa ordem, pois nem todas as tarefas possuem relação de precedência com as tarefas posteriores. É normal ocorrer muito “vai e vem” na fase de desenvolvimento para que as funcionalidades sejam testadas e evoluam.

Ainda, mais do que dizer que a ordem das tarefas não é única, esse passo a passo em si é apenas uma das praticamente infinitas possibilidades de se desenvolver um software que resolva um problema específico ou sirva como um MVP. Ele é uma opção simples, rápida de ser desenvolvida e fácil de ser aprendida. Um bom conselho quando se trata de MVP e de colocar um negócio em prática é o famoso “feito é melhor que perfeito”. Não existe um consenso a respeito de qual a melhor solução, qual o melhor conjunto de tecnologias (*stack*), qual a melhor linguagem, etc. O melhor, especialmente para um MVP, é testar e aprender da forma mais barata, simples e rápida possível, desde que seja realista para o estágio de testes que se faz necessário para as validações das hipóteses atuais. Em outras palavras, use e adapte esse passo a passo da forma como achar mais conveniente, caso já tenha alguma experiência com outra tecnologia ou ferramenta, mas não gaste tempo tentando encontrar a melhor resposta. Ela não existe.

Falando nisso, o código aqui desenvolvido não usa o paradigma de orientação a objetos e nem está organizado no padrão MVC (*Model – View – Controller*), pois o *backend* não produz *views* e o *frontend* não lida com *models*, entre outras coisas. A

saber, a comunicação entre *front* e *back* é baseada em JSONs, as *views* são manipuladas apenas no *front* e a conversa com os modelos de dados apenas no *back*. É um modelo embrionário de API, porém também não é REST e nem está organizada em micro serviços. O ponto é que deve-se levar em conta quando, e se, vale a pena ter o *overhead* (custo) de organização do código em padrões “de mercado”. Não é pecado, por exemplo, iniciar um produto/serviço com um “monólito” e depois evoluí-lo quando necessário. O importante é ter uma forma de organizar o código inicial de modo que ele faça sentido para o desenvolvedor, ou pequena equipe, do MVP.

Uma curiosidade sobre o código desenvolvido aqui é que foram escritas 152 linhas de código para o *app*, 131 linhas para o *backend* e 175 linhas para a parte administrativa na web, totalizando apenas 458 linhas de código para todo o MVP. Esse número considera linhas que possuem alguma função, mesmo que apenas um fechamento de chaves.

Finalmente, ao longo de todo este guia existe um grande *trade-off* entre deixá-lo sucinto a ponto de ser realmente útil e aprofunda-lo em detalhes para transmitir o máximo de conhecimento possível. Mesmo assim, apesar de extenso, na maior parte do tempo venceu a simplicidade e até mesmo a superficialidade em alguns aspectos para que o objetivo de servir como guia prático para o desenvolvimento de um MVP fosse atingido com maior sucesso. Portanto, recomenda-se que para algo além de um MVP, o leitor estude sobre prós e contras dessa alternativa apresentada aqui, estude outras alternativas, estude questões como segurança, performance, limitações, entre outras, e atente-se para as licenças de cada software utilizado.

## 5. Desenvolvendo o MVP na prática

### 5.1. Setup inicial do ambiente de desenvolvimento local

Baixe e instale em sua máquina o XAMPP, o Google Chrome, o FileZilla, o Workbench e o Sublime Text ou um editor de texto de sua preferência. Esta etapa não deve apresentar grandes dificuldades, pois as configurações padrão desses produtos, geralmente, funcionam perfeitamente e não será preciso nenhuma customização especial. Portanto, não há um passo a passo detalhado e nem há imagens nessa etapa.

### 5.2. Contratação e configuração do servidor

Esta etapa não precisa, necessariamente, ser feita com grande antecedência, caso o desenvolvimento do seu MVP vá levar algumas semanas ou meses. Porém, como faremos um MVP em poucas horas, já vamos contratar o servidor, pois precisaremos saber quais são seus *Name Servers* para poder apontar o DNS do nosso domínio para eles. Como a propagação das mudanças de DNS pode demorar horas, é bom antecipar essa etapa.

Os Servidores de Nome de Domínio (em inglês: *Domain Name Servers - DNS*), são responsáveis pela resolução dos nomes de internet, ou seja, são eles que transformam nomes legíveis em endereços de IP roteáveis. Na próxima etapa, quando formos registrar um domínio, será preciso dizer para o *Registrar*, entidade que faz o registro de domínios, quais são os nomes dos *Name Servers* do servidor que iremos utilizar, pois é lá que o nosso código de *backend* estará hospedado fisicamente e é para lá que as requisições dos nossos usuários precisam ser direcionadas.

Existem centenas de opções de servidores que poderiam atender as nossas necessidades, inclusive alguns muito robustos que possuem *free-tiers*, como a Amazon, que permite que rodemos aplicações simples em pequenos servidores por até um ano de graça. O *trade-off* aqui consiste em encontrar um bom balanço entre qualidade, preço e complexidade para usar. A Amazon, por exemplo, com sua *free-tier* ofereceria uma ótima qualidade gratuitamente por um ano, porém, por ser uma opção muito robusta e sofisticada, ela exige mais configuração, e isso não faz parte do escopo deste guia. Portanto, julgou-se uma opção mais adequada aquela que oferecesse boa qualidade e o mínimo de configuração, a um preço baixo.

Uma boa alternativa que cumpre com esses requisitos é o produto *Shared Host*, do Dreamhost (<https://www.dreamhost.com/hosting/shared/>). O autor do guia possui uma conta já com créditos nesse servidor e recomenda-o para esse tipo de uso. Portanto, siga o link acima, veja a lista de funcionalidades, a precificação e contrate, caso esteja satisfeito e queira seguir o passo a passo como apresentado aqui.

Após contratado o serviço, passo que não cobriremos neste guia, será preciso realizar uma pequena etapa de configuração. Certifique-se de estar logado no painel de controle de sua conta, vá então até “*Manage Domains*” dentro de “*Domains*”, no menu lateral esquerdo, conforme figura 2 a seguir:

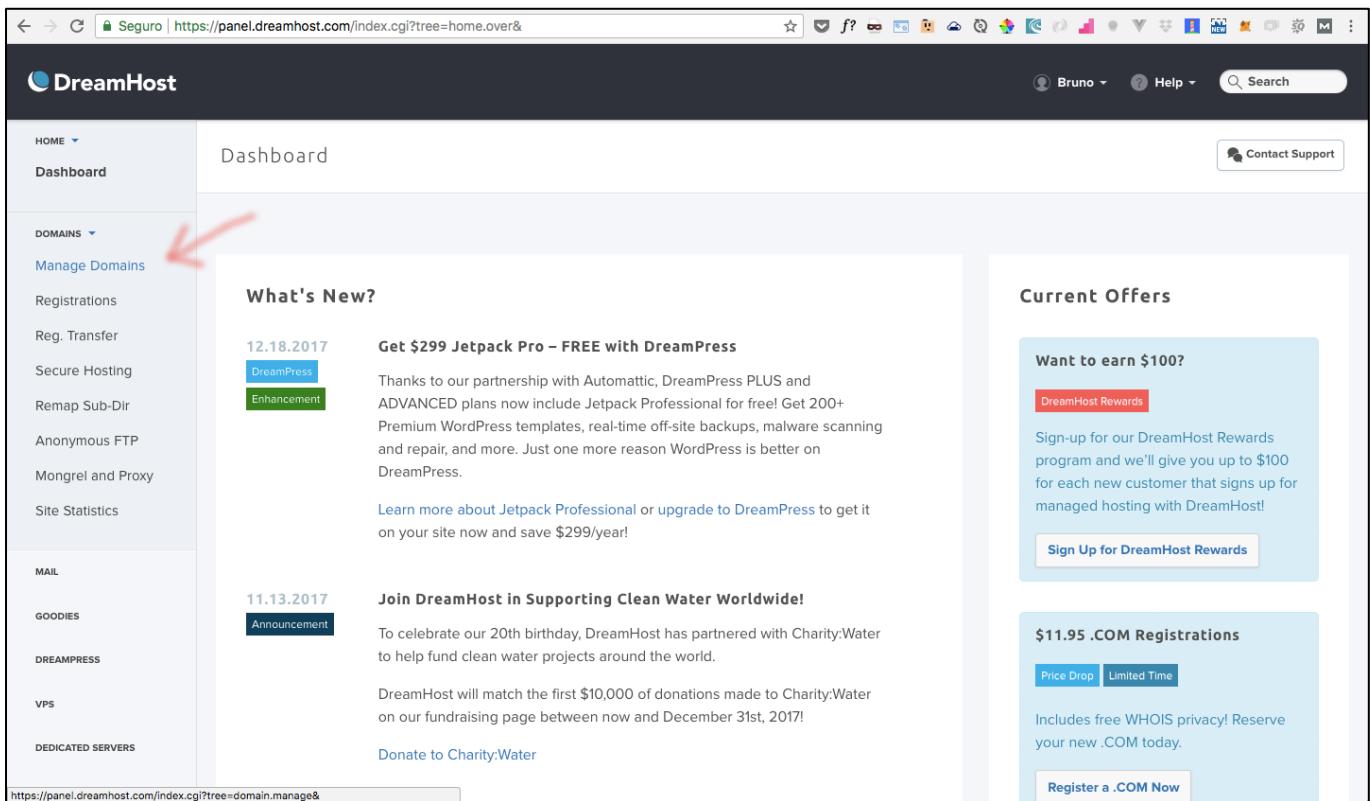


Figura 2 - Dashboard Dreamhost: Manage Domains

Em seguida, clique em “*Add hosting to a Domain / Sub-Domain*”, conforme figura 3:

The screenshot shows the DreamHost control panel under the 'Manage Domains' section. A red arrow points to the 'Add Hosting to a Domain / Sub-Domain' button. The interface includes a sidebar with various service links like HOME, DOMAINS, MAIL, GOODIES, DREAMPRESS, VPS, DEDICATED SERVERS, REMIXER, and CLOUD SERVICES. The main area displays a table of hosted domains with columns for Domain, Registration, Web Hosting, Security, Email, and Actions.

Domain	Registration	Web Hosting	Security	Email	Actions
[REDACTED]	Unknown	Edit   Deactivate redirect	https Off	Not us!	<span>✗ Delete</span>
[REDACTED]	Unknown	Fully Hosted with PHP 5.6 [REDACTED] Edit   Remove	https Off Malware Removal Off	Not us!	<span>⟳ Restore</span> <span>✗ Delete</span>
[REDACTED]	Unknown	Edit   Deactivate redirect	https Off	Not us!	<span>✗ Delete</span>
[REDACTED]	Unknown	Edit   Deactivate redirect	https Off	0 Addresses	<span>✗ Delete</span>
[REDACTED]	Unknown	CloudFlare + Fully Hosted with PHP 5.6 [REDACTED] Edit   Remove	https Off Malware Removal Off	0 Addresses	<span>⟳ Clear Cache</span> <span>⟳ Restore</span> <span>✗ Delete</span>
[REDACTED]	Unknown	CloudFlare + Fully Hosted with PHP 7.0 [REDACTED]	https On Certificate Active	Not us!	<span>⟳ Clear Cache</span>

Figura 3 – Add hosting to a Domain / Sub-Domain

Na tela que se abriu, coloque o endereço de domínio em “*Domain to host*”. No nosso caso, o endereço é [formiga.info](http://formiga.info). Certifique-se de que o domínio esteja disponível para compra, ou até mesmo já o compre antes dessa etapa (avance para a etapa 5.3 para mais informações).

Escolha se você quer adicionar “www” em todas as requisições, se quer remover ou se quer deixar sempre da forma como ela for feita. Para fins de MVP, isso não é relevante.

Crie um novo usuário (recomendado) para poder acessar os arquivos deste domínio, não esqueça de anotar em um lugar seguro e não compartilhar essa informação com ninguém. Não é necessário alterar o diretório web e nem a opção de PHP. Veja a figura 4, a seguir:

The screenshot shows the DreamHost panel interface. On the left, there's a sidebar with various service links like HOME, DOMAINS, MAIL, GOODIES, DREAMPRESS, VPS, DEDICATED SERVERS, REMIXER, and CLOUD SERVICES. The main area is titled 'Manage Domains' and has a sub-section for 'Fully Hosted'. It asks 'Upload your site to our servers and we'll serve it up!'. There are fields for 'Domain name' (set to 'formiga.info'), 'Domain to host' (set to 'sub-domains are okay!'), and 'Do you want the www in your URL?' with three radio button options: 'Leave it alone: Both http://www.formiga.info/ and http://formiga.info/ will work.', 'Add WWW: Make http://formiga.info/ redirect to http://www.formiga.info/ (selected)', and 'Remove WWW: Make http://www.formiga.info/ redirect to http://formiga.info/'. Below this, there are sections for 'Users, Files, and Paths' (with a note about running the domain under a user), 'Web Options' (with a dropdown for PHP mode set to 'PHP 7.0 FastCGI (Default)'), and 'Logs directory' (set to '/home/username/logs/formiga.info/http'). A 'Create a New User (Recommended)' button is also visible.

Figura 4 – Configurando um novo domínio, parte 1

Dando continuidade à configuração, adicione um certificado SSL grátis e escolha obter “Extra Web Security”. O upgrade automático do PHP é opcional para nosso caso e o CloudFlare, um serviço de CDN, não é necessário para o MVP. A saber: CDN significa *Content Distribution Network*, isso é uma rede mundial de distribuição de conteúdo que serve, principalmente, para deixar seu serviço mais rápido, uma vez que os arquivos estáticos (imagens, documentos, etc.) estarão mais próximos do local onde serão consumidos. Por exemplo, se o seu servidor está localizado nos EUA (nossa caso) e o usuário do seu app está no Brasil, o CloudFlare guardará cópias dos arquivos estáticos, por exemplo, em São Paulo e os fornecerá para seu usuário aqui no Brasil a partir deste servidor, economizando tempo, uma vez que os dados viajarão menos. Também não é necessário marcar a caixa para Google Apps, pois não vamos contratar esse serviço agora. Feito isso, clique em “*Fully host this domain*”, conforme figura 5 a seguir:

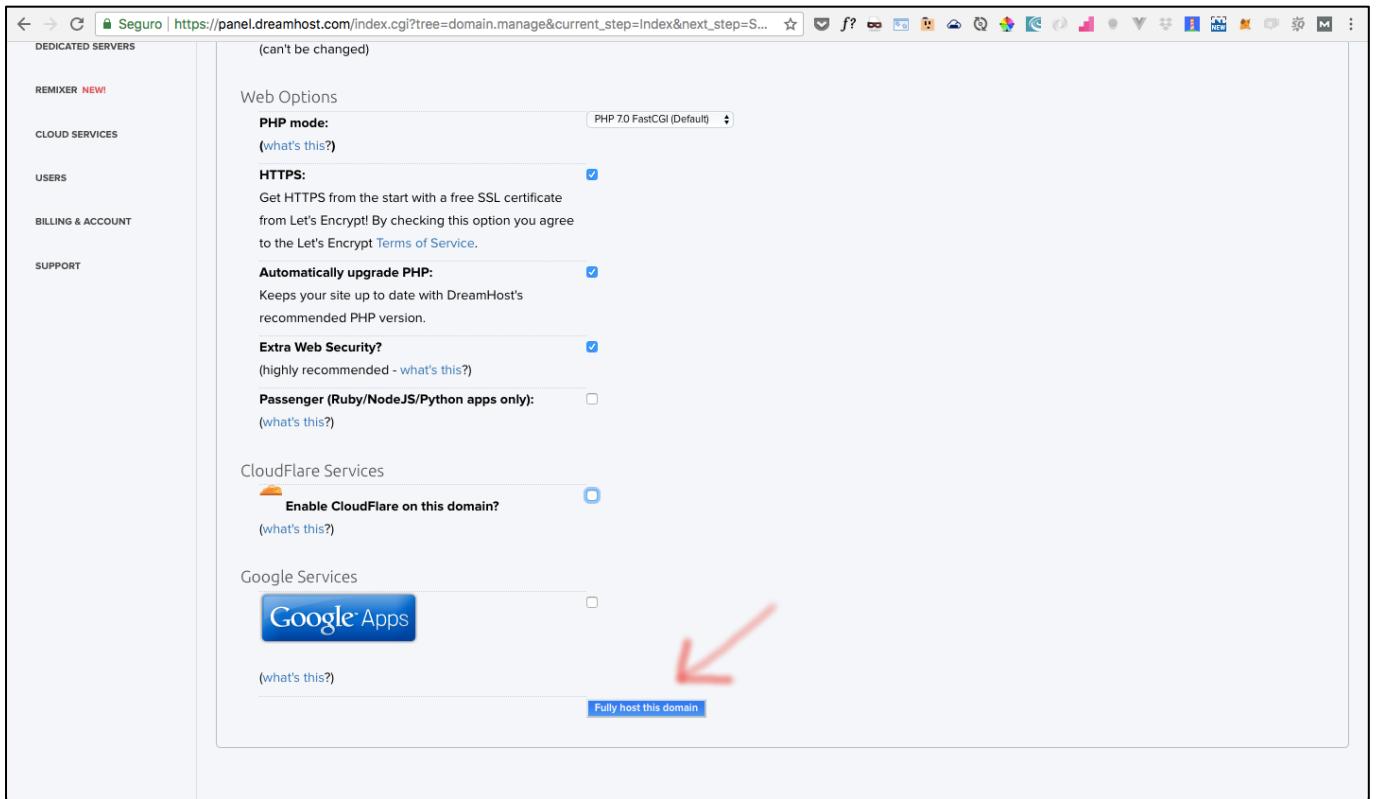


Figura 5 – Configurando um novo domínio, parte 2

Se tudo correr bem, você deverá ser redirecionado para a tela anterior (“Manage Domains”) e ver um aviso em verde, conforme a figura 6 a seguir. Lembre-se de anotar a senha de acesso para o usuário que você acabou de criar em algum local seguro e não compartilhe-a com ninguém. Se quiser, é possível alterar essa senha para algo mais amigável. Note que ao final do aviso estão os *Name Servers*: NS1.DREAMHOST.COM, NS2.DREAMHOST.COM e NS3.DREAMHOST.COM. São esses nomes que informaremos ao *Registrar* mais a frente.

The screenshot shows the DreamHost control panel interface. On the left, a sidebar lists various services: HOME, DOMAINS (selected), Manage Domains, Registrations, Reg. Transfer, Secure Hosting, Remap Sub-Dir, Anonymous FTP, Mongrel and Proxy, Site Statistics, MAIL, GOODIES, DREAMPRESS, VPS, DEDICATED SERVERS, REMIXER (NEW!), and CLOUD SERVICES. The main content area is titled "Manage Domains". A green "Success!" box states: "'formiga.info' has been added to our hosting system! Since it takes some time for new DNS information to propagate, it may take up to a few hours for your new domain to start working. Right now no email addresses have been set up yet @formiga.info. You can do that from our 'Mail > Manage Email' area at any time. Within ten minutes the new SFTP user [REDACTED] will be created with password [REDACTED]... if you'd like to change the name, shell type, password, or anything else about this user, feel free to do so [here!](#)". Below this, under "MAIL", there is a table with three rows: NS1.DREAMHOST.COM, NS2.DREAMHOST.COM, and NS3.DREAMHOST.COM, each followed by an IP address (64.90.62.230, 208.97.182.10, and 66.33.205.230). A red arrow points to the first row. At the bottom of the main area, it says "Thanks for choosing us, The Happy DreamHost Domain Adding Robot!". Two blue buttons are present: "Add Hosting to a Domain / Sub-Domain" and "Register a New Domain". Below these buttons, it says "All hosted domains on this account" and "Showing 1 sub-domain. Hide them?".

Figura 6 – Sucesso ao configurar um novo domínio

Feita a configuração do domínio, vamos aproveitar o painel de controle aberto para criar uma nova base de dados a ser utilizada pelo formiga.info. Vá em “MySQL Databases”, dentro de “Goodies”, no menu lateral esquerdo, conforme figura 7 a seguir:

The screenshot shows the DreamHost Dashboard. On the left sidebar, under the 'GOODIES' category, 'MySQL Databases' is listed with a red arrow pointing to it. The main content area features a 'What's New?' section with two items:

- 12.18.2017** Get \$299 Jetpack Pro – FREE with DreamPress
 

Thanks to our partnership with Automattic, DreamPress PLUS and ADVANCED plans now include Jetpack Professional for free! Get 200+ Premium WordPress templates, real-time off-site backups, malware scanning and repair, and more. Just one more reason WordPress is better on DreamPress.

[Learn more about Jetpack Professional](#) or [upgrade to DreamPress](#) to get it on your site now and save \$299/year!
- 11.13.2017** Join DreamHost in Supporting Clean Water Worldwide!
 

To celebrate our 20th birthday, DreamHost has partnered with Charity:Water to help fund clean water projects around the world.

DreamHost will match the first \$10,000 of donations made to Charity:Water on our fundraising page between now and December 31st, 2017!

[Donate to Charity:Water](#)

On the right side, there are 'Current Offers' sections for 'Want to earn \$100?' (DreamHost Rewards) and '\$11.95 .COM Registrations' (Price Drop, Limited Time).

Figura 7 – Dashboard Dreamhost: MySQL Databases

Dentro de *MySQL Databases*, clique em “Add New Hostname”, conforme figura 8:

The screenshot shows the 'MySQL Databases' page. The left sidebar has 'MySQL Databases' selected. The main content area includes a 'MySQL Recycle Bin' notice and a 'MySQL Server' section with a link to 'Turn on a MySQL VPS!'. Below this, there is a form titled 'Hostnames for this MySQL server:' with a blue 'Add New Hostname' button, which is highlighted with a red arrow. A table lists hostnames with their corresponding Web Administration links (phpMyAdmin) and delete actions.

Hostname (these are all interchangeable)	Web Administration	Actions
[REDACTED]	phpMyAdmin	<input type="button" value="Delete Hostname"/>
[REDACTED]	phpMyAdmin	<input type="button" value="Delete Hostname"/>
[REDACTED]	phpMyAdmin	<input type="button" value="Delete Hostname"/>
[REDACTED]	phpMyAdmin	<input type="button" value="Delete Hostname"/>
[REDACTED]	phpMyAdmin	<input type="button" value="Delete Hostname"/>
[REDACTED]	phpMyAdmin	<input type="button" value="Delete Hostname"/>

Figura 8 – Add New Hostname

Dê um nome para o novo *Hostname*, selecione o domínio correspondente no *dropdown* ao lado e, por fim, clique em “*Create this MySQL hostname now!*”, conforme figura 9:

The screenshot shows the DreamHost panel interface. On the left is a sidebar with various links: HOME, DOMAINS, MAIL, GOODIES (with MySQL Databases selected), One-Click Installs, Subversion, Cron Jobs, Jabber IM, Htaccess/WebDAV, Block Spiders, DREAMPRESS, VPS, DEDICATED SERVERS, REMIXER NEW!, CLOUD SERVICES, and USERS. The main content area is titled "MySQL Databases" and shows a form for adding a MySQL Hostname. It includes fields for "New Hostname" (containing a redacted domain name), a dropdown for "Domain", and a note: "This domain must be registered and use our DNS (name servers)!". Below the form is a message about DNS updates and a "Create this MySQL hostname now!" button.

Figura 9 – Criando um novo hostname

Com o novo Hostname criado, você retornará à tela anterior. Agora, é hora de criar uma nova base de dados. Role a tela “*MySQL Databases*” para baixo até a seção “*Create a new MySQL database*”. Dê um nome para a nova base de dados, selecione o hostname que você acabou de criar no dropdown, escolha o usuário, defina e confirme uma senha para acesso desse usuário à base de dados, insira algum comentário a respeito dessa base (opcional e para fins de organização apenas) e, por fim, clique em “*Add new database now!*”, conforme figura 10:

Seguro | https://panel.dreamhost.com/index.cgi?tree=goodies.mysql&

"MySQL 5 Database" 2016-11-23 09:22:47

**Create a new MySQL database:**

**Database Name:** [REDACTED]

Try putting your domain name in front of your database name if the name you want is already taken.

**Use Hostname:** [REDACTED]

**New Hostname:** [REDACTED] This domain must be registered and use our DNS (name servers)!

There is a delay while DNS updates for new hostnames (this can take up to a few hours). In the meantime, feel free to use any of your existing hostnames on the same service to connect to your database.

**First User:** Create a new user now... [REDACTED]

**New Username:** [REDACTED]

**New Password:** ..... Strong

Must be at least 8 characters

**New Password Again:** .....

**Database Comment:** DB Formiga.info

Optional - for your own organizational use!

Your database will be created right away, however new hostnames will need time to propagate.

**Add new database now!** 

Figura 10 – Criando uma nova base de dados

Se tudo correr bem, você deverá ser redirecionado para a página "MySQL Databases" e ver um alerta verde similar ao da figura 11:

Seguro | https://panel.dreamhost.com/index.cgi?

Bruno Help Search Contact Support

**MySQL Databases**

**MySQL Recycle Bin**

(Restore a database you deleted before it gets purged from the recycle bin.  
Purging can take 30 or more days, during which you will be unable to reuse the same database name.)

**Success!**

Your database [REDACTED] has been created!

Your hostname [REDACTED] will be set up within about 5-10 minutes.  
You MUST always use your hostname to connect to your database... "localhost" WILL NOT WORK.

Connect to your new database from the command line with:  
`mysql -u [REDACTED] -p -h [REDACTED]`  
You can also go to [REDACTED] to manage your MySQL database from the web (once the hostname is set up of course).

**MySQL Server**

Need more SQL power, customization or security? Turn on a MySQL VPS!

Hostnames for this MySQL server:

Add New Hostname

Figura 11 – Sucesso ao criar uma nova base de dados

A mensagem desse alerta vai variar caso você tenha criado o novo *hostname* juntamente com a base de dados ou separadamente. De qualquer forma, feito isso, precisamos agora autorizar nosso endereço IP a acessar essa base de dados para que possamos criar as tabelas necessárias. Vale lembrar que o novo *hostname* pode levar alguns minutos para começar a funcionar.

Para autorizar seu IP, role a tela “MySQL Databases” até encontrar a parte que diz “*Database(s) on this server:*” e então clique no usuário correspondente à base de dados que você criou, conforme figura 12 a seguir:

Database	Description	Last Action	Users with Access	Actions
[REDACTED]	[REDACTED]	2017-06-28 19:11:35	[REDACTED]	<a href="#">Add a user</a> <a href="#">Restore DB</a> <a href="#">Modify DB</a> <a href="#">Delete DB</a>
[REDACTED]	"BD formiga.info"	2018-05-02 18:18:07	[REDACTED]	<a href="#">Add a user</a> <a href="#">Restore DB</a> <a href="#">Modify DB</a> <a href="#">Delete DB</a>
[REDACTED]	[REDACTED]	2016-11-23 09:05:22	[REDACTED]	<a href="#">Add a user</a> <a href="#">Restore DB</a> <a href="#">Modify DB</a> <a href="#">Delete DB</a>
[REDACTED]	[REDACTED]	2016-11-23 09:34:08	[REDACTED]	<a href="#">Add a user</a> <a href="#">Restore DB</a> <a href="#">Modify DB</a> <a href="#">Delete DB</a>

Figura 12 – Permissões de acesso para o usuário, parte 1

Na próxima tela, encontre o local que diz “*What may \*user\* do to tables in these databases?*”. Ao final dessa seção, você verá uma linha dizendo “*Your current computer is: \*IP\**”. Esse é o IP do seu computador atual. Copie esse IP e insira-o no *textbox* ao lado, na linha de baixo de “%.dreamhost.com”, sem alterar essa linha. Clique no botão “*Modify \*user\* now!*”.

The screenshot shows a web-based MySQL configuration interface. On the left, a sidebar lists various services: One-Click Installs, Subversion, Cron Jobs, Jabber IM, Htaccess/WebDAV, Block Spiders, DREAMPRESS, VPS, DEDICATED SERVERS, REMIXER NEW!, CLOUD SERVICES, USERS, BILLING & ACCOUNT, and SUPPORT. The main area is titled "Database Name" and shows a list of actions with checkboxes: Select (checked), Insert (checked), Update (checked), Delete (checked), Create (checked), Drop (checked), Index (checked), and Alter (checked). Below this, under "Allowable Hosts", it says "From what hosts (computers) may [REDACTED] connect to these databases? (One per line, use % as a wildcard.)" and shows "%dreamhost.com". It also displays "Your current computer is: [REDACTED]". A section titled "Do you need to know [REDACTED] password?" has fields for "Current Password" (with a redacted value) and "Show". Another section titled "Would you like to change [REDACTED] password?" has fields for "New Password" and "New Password Again", both with redacted values. A blue button labeled "Modify [REDACTED] now!" is at the bottom.

Figura 13 – Permissões de acesso para o usuário, parte 2

Pronto! O servidor já está configurado para receber o código e já é possível acessá-lo com o workbench para criar as tabelas.

### 5.3. Comprando um domínio e configurando o DNS

É nessa fase que precisamos batizar nosso projeto (lembrando que uma pesquisa prévia de disponibilidade deve ter sido feita antes da configuração do servidor).

Existem muitos *Registrars*, empresas que realizam o registro de domínios. Em nosso exemplo, vamos utilizar o Google Domains (<https://domains.google/#/>). Vá até este endereço, digite na caixa de busca o nome do domínio que você deseja comprar e dê um *enter*.

Aparecerá uma página com os resultados mais relevantes, assim como algumas sugestões de domínios baseadas na sua busca. Se o domínio estiver disponível, aparecerá um valor anual e, caso contrário, aparecerá um símbolo indicando indisponibilidade. Encontre um domínio disponível, adicione-o ao seu carrinho, preencha todos os dados necessários e realize a compra. Será preciso confirmar seu e-mail.

Algumas dicas na hora de escolher um nome para seu negócio:

- Pense em qual vai ser a abrangência dele, é apenas local? É nacional? Internacional? Caso haja a possibilidade de internacionalização no futuro, talvez seja interessante baseá-lo num domínio “.com”, por exemplo, para facilitar a memorização e evitar perda de tráfego.
- Se você optar por um domínio .com, pense se não vale a pena também comprar o “.com.br” para, da mesma forma, evitar confusão, perda de tráfego ou até mesmo para se proteger contra um eventual problema com a sua marca, uma vez que o negócio se iniciará no Brasil.
- Veja se o nome escolhido é fácil de ser pronunciado, por exemplo, numa conversa telefônica, ou se é necessário explicar / soletrar.
- Veja também se esse nome não possui nenhum significado indesejado em alguma das possíveis línguas que seu negócio possa vir a atuar.
- Não compre “qualquer um para depois trocar por um definitivo”, pois conforme você for gerando conteúdo e o Google começar a ranquear as suas URLs, você terá mais trabalho depois para conseguir transferir essa autoridade para o novo domínio. Sem contar que a autoridade off-line da sua marca também será perdida.
- Pense em algo criativo, pesquise bastante e divirta-se!

No caso desse nosso exemplo, escolhemos o domínio “formiga.info”, pois o projeto se trata de um serviço de comunicação e distribuição de informações, e as formigas são insetos famosos por se comunicarem ☺.

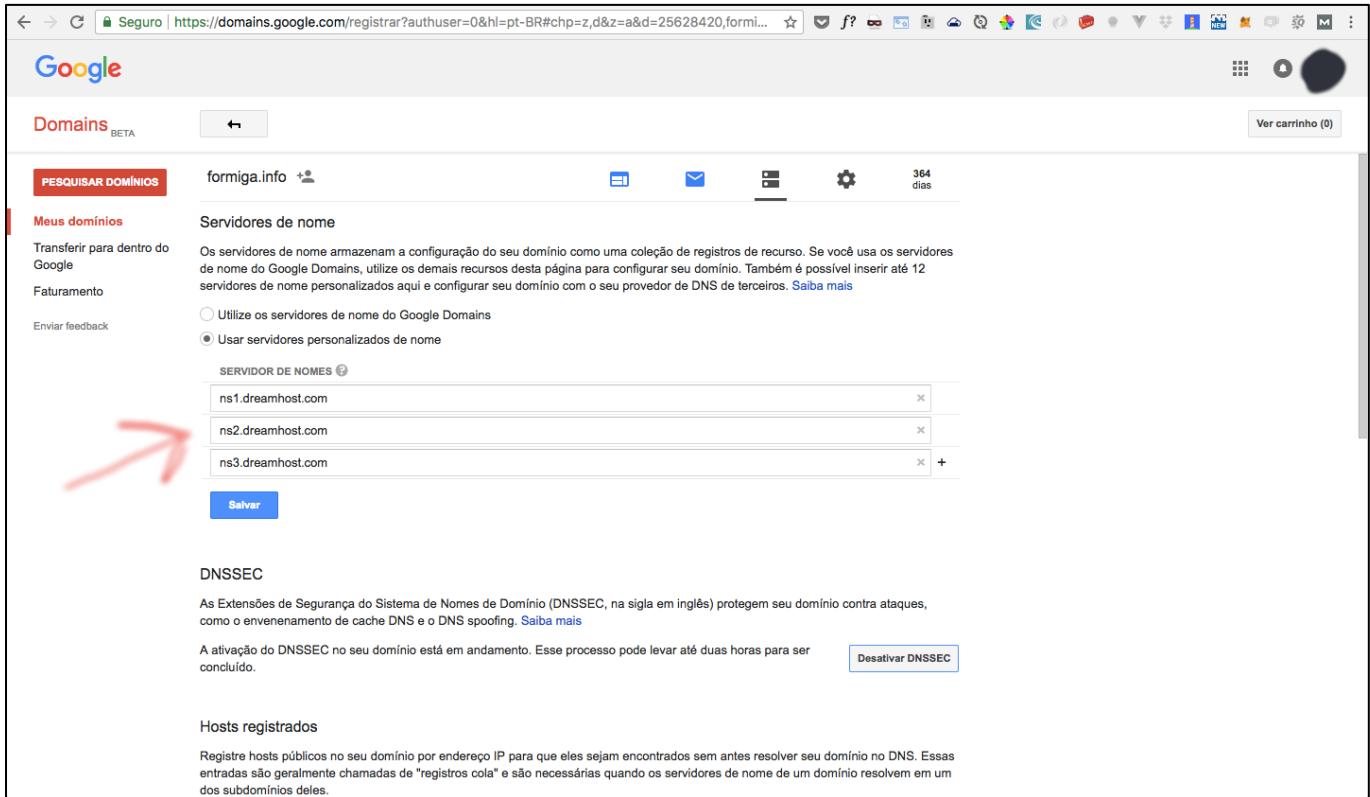
Comprado o domínio, clique no botão do DNS para configura-lo, conforme figura 14:

The screenshot shows the Google Domains BETA interface. At the top, there's a navigation bar with links for Seguro, Domains BETA, and Ver carrinho (0). Below the header, there's a sidebar with sections for PESQUISAR DOMÍNIOS, Meus domínios (which is highlighted in red), Transferir para dentro do Google, Faturamento, and Enviar feedback. The main content area displays a table with columns: DOMÍNIO, WEBSITE, E-MAIL, DNS (with a red arrow pointing to it), CONFIGURAÇÕES, and EXPIRA EM. The row for 'formiga.info' shows the website icon, email icon, a gear icon for DNS, and a 364 dias expiration date. Below the table, there's a message: 'Você está conectado como [redacted]. Não consegue encontrar seu domínio? Encontre ajuda aqui.' At the bottom of the page, there are links for Sobre, Recursos, Ajuda, Idioma: português do Brasil, and País legal/de faturamento: Brasil. The footer contains the text '©2018 Google - Termos de Serviço - Política de privacidade'.

Figura 14 – Configurando o DNS, parte 1

Chegou a hora de usar aqueles *Name Servers* que vimos há alguns passos atrás.

Insira-os conforme a figura 15, a seguir, e clique em “Salvar”:



The screenshot shows the Google Domains BETA interface for managing the domain 'formiga.info'. In the 'Servidores de nome' (Name Servers) section, there is a list of three custom name servers: 'ns1.dreamhost.com', 'ns2.dreamhost.com', and 'ns3.dreamhost.com'. A red arrow points to this list. Below the list is a blue 'Salvar' (Save) button. At the bottom of the page, there is a 'DNSSEC' section with a 'Desativar DNSSEC' (Disable DNSSEC) button.

Figura 15 – Configurando o DNS, parte 2

Pronto, a configuração do DNS está completa! Porém, lembre-se que ela pode levar algumas horas até propagar e ficar 100% operacional. No caso do Dreamhost, ele cria automaticamente uma página para domínios hospedados por ele que ainda estão em construção. Então, a cada 10 ou 15 minutos visite o seu domínio para verificar a situação do DNS. Caso você encontre uma página igual a apresentada na figura 16, significa que, provavelmente, o DNS já terminou de propagar.

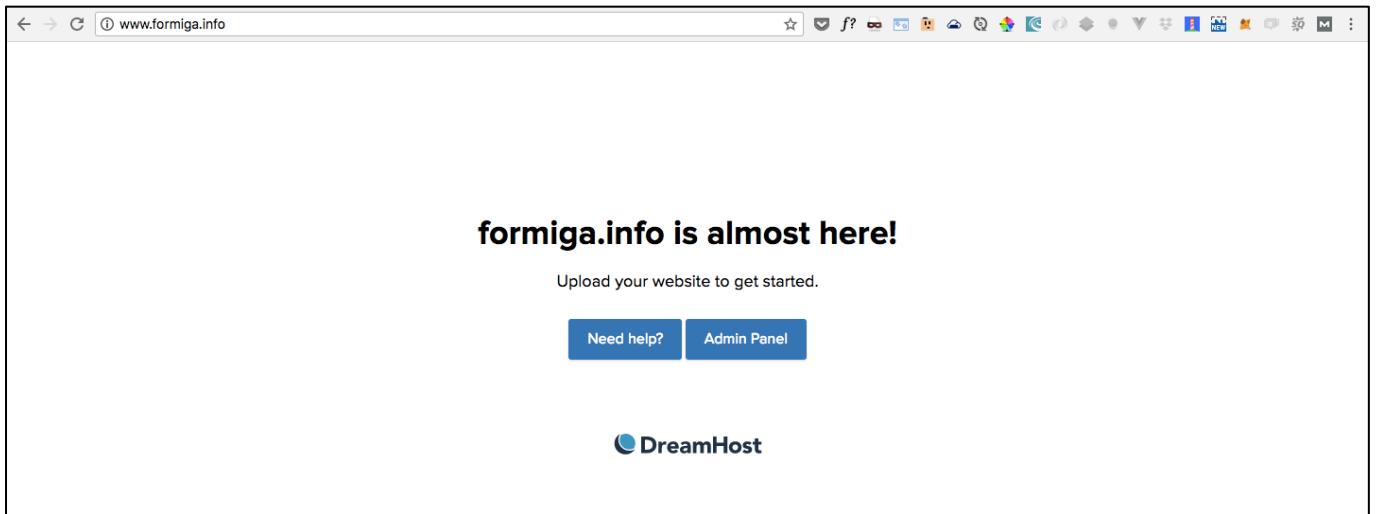


Figura 16 – DNS configurado com sucesso

#### 5.4. Criando as conexões com o banco de dados local e de produção

Abra o Workbench e clique no ícone de nova conexão, na tela inicial, conforme a figura 17. Certifique-se de estar na aba de conexões, a do golfinho ao lado esquerdo.

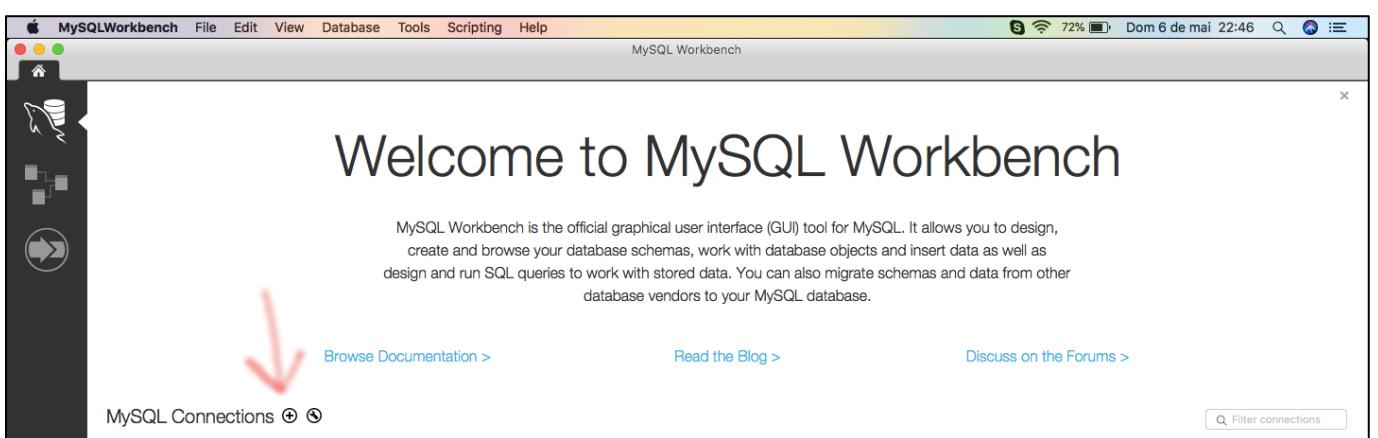


Figura 17 – Criando uma nova conexão com o banco de dados, parte 1

Primeiramente, vamos criar uma conexão com o banco de dados local. Para isso, abra o XAMPP e certifique-se de que o serviço *MySQL Database* esteja rodando, conforme figura 18. Aproveite para já ativar o *Apache Web Server*, caso ele ainda não esteja rodando, pois precisaremos dele mais para a frente.

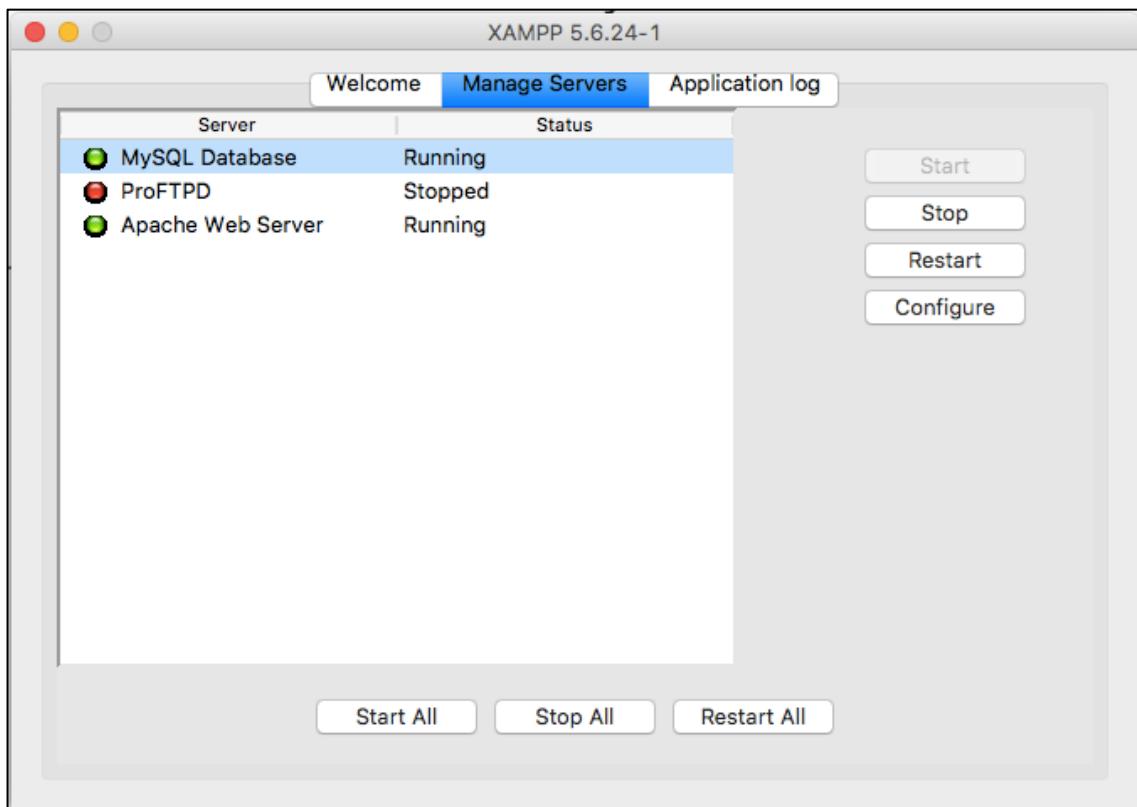


Figura 18 – XAMPP: MySQL Database + Apache Web Server

Se estiver tudo certo, volte para o Workbench e dê um nome para a conexão local, por exemplo: localhost. Se você não alterou nenhuma configuração do XAMPP, não será necessário mexer nos demais campos da conexão. O *Hostname* será seu IP, 127.0.0.1, a porta será a 3306, o *Username* será root e a *Password* será vazia. Clique em “OK” e a conexão com o banco local estará criada. Veja o exemplo na figura 19.

É válido ressaltar aqui que existem formas mais seguras de realizar essa conexão com o banco de dados, principalmente, de produção. Por exemplo, selecionando SSL como modo de conexão. Para a conexão com o banco de dados de produção, recomenda-se utilizar alguma forma mais segura. Porém, isso não será coberto neste guia. Pesquise a respeito para entender quais são os riscos de uma conexão comum e como fazer uma conexão mais segura.

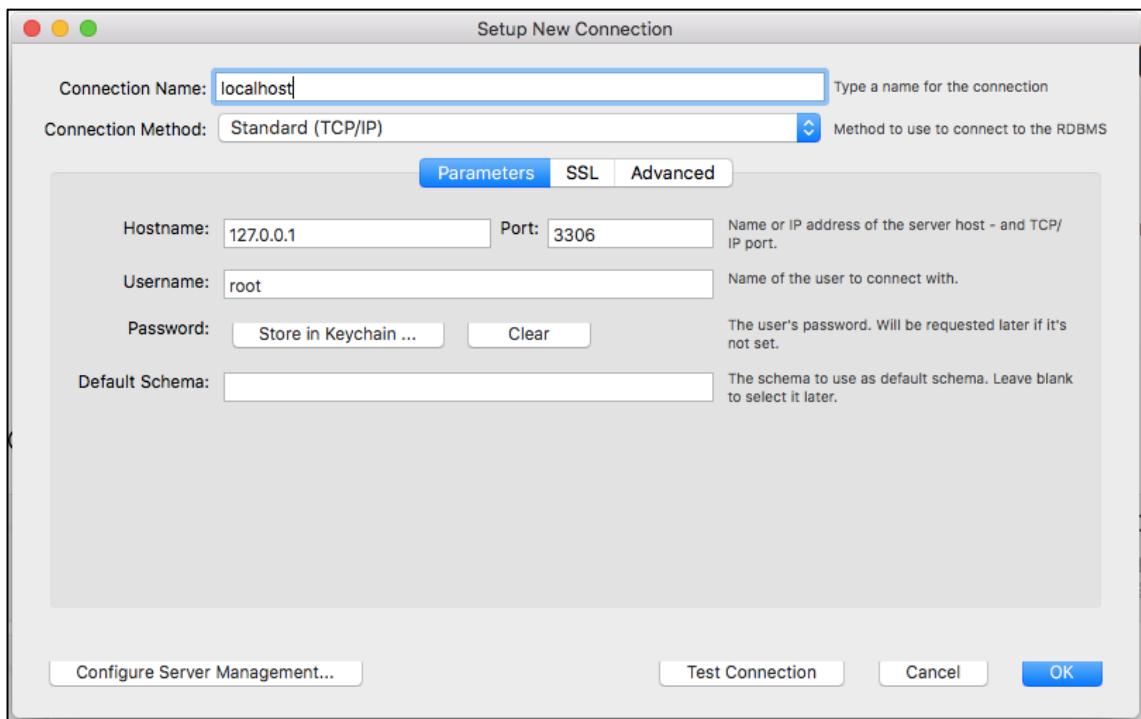


Figura 19 – Criando uma nova conexão com o banco de dados, parte 2

Repita o processo agora para criar uma conexão com o banco de dados do nosso servidor de produção. Substitua o *hostname* pelo criado na figura 9 e o *username* e *password* pelos criados na figura 10. Clique em “OK”. Se tudo correr bem, agora você terá dois atalhos na tela inicial do workbench, um para cada uma das conexões, conforme figura 20:

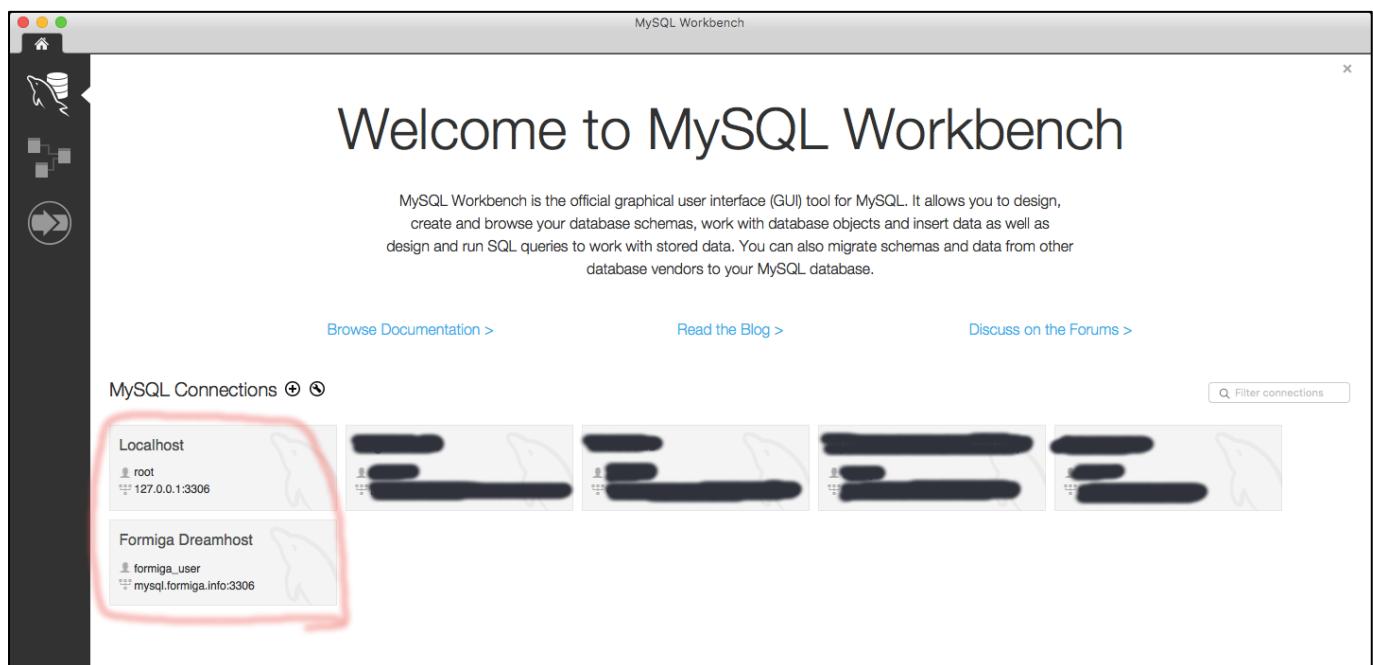


Figura 20 – Criando uma nova conexão com o banco de dados, parte 3

## 5.5. Criando um novo *schema* para o banco de dados

Agora que as conexões estão criadas, vamos criar o *schema* do banco de dados, para que as tabelas sejam criadas. Certifique-se de estar na aba com a “árvore de pastas” do lado esquerdo e, então, clique no botão de novo *schema*, conforme figura 21:

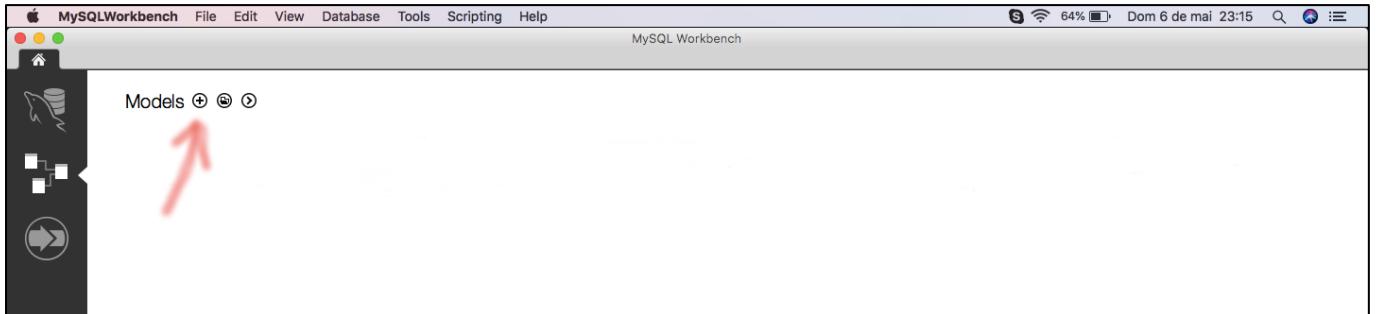


Figura 21 – Criando um novo schema, parte 1

A seguir, dê um clique duplo em “Add Diagram”, conforme figura 22:

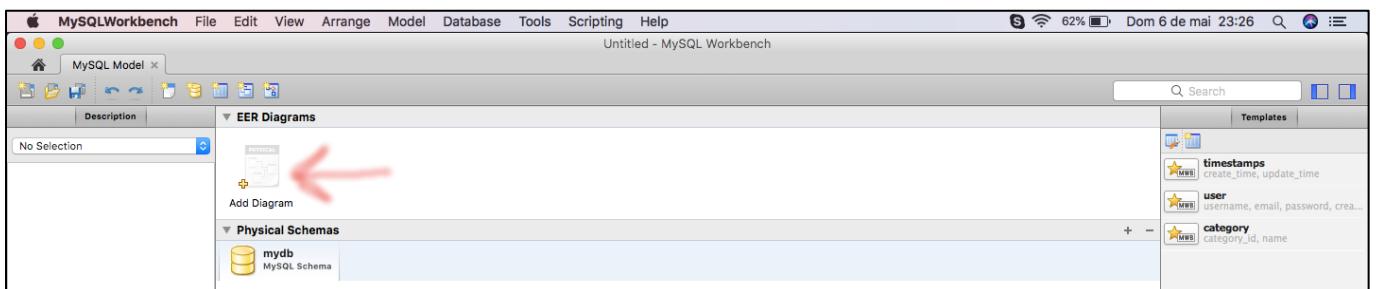


Figura 22 – Criando um novo schema, parte 2

Dentro do editor de *schemas*, clique em “Place a new table”, conforme figura 23:

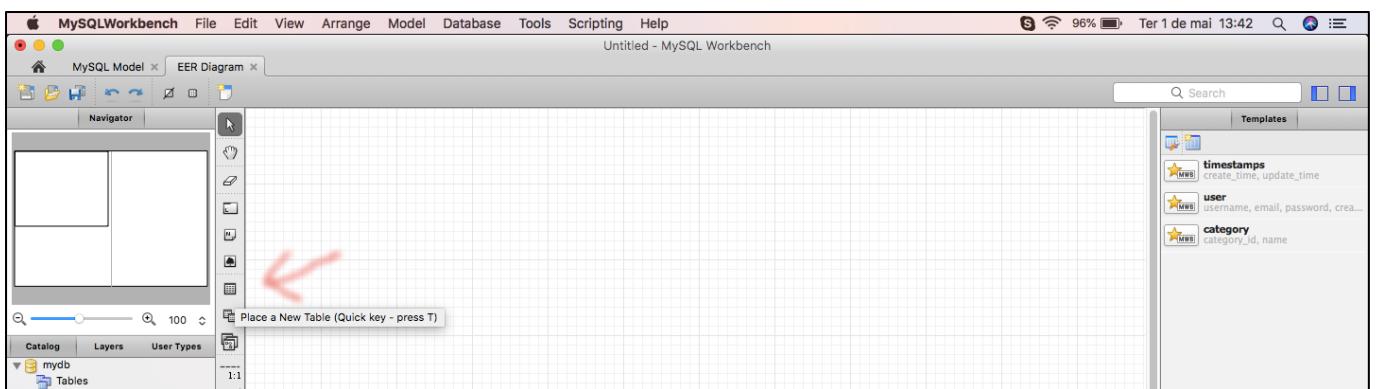


Figura 23 – Criando um novo schema, parte 3

Coloque a nova tabela sobre a folha em branco e dê um duplo clique nela para abrir o modo de edição, conforme figura 24. A nova tabela terá o nome de “table1”, inicialmente.

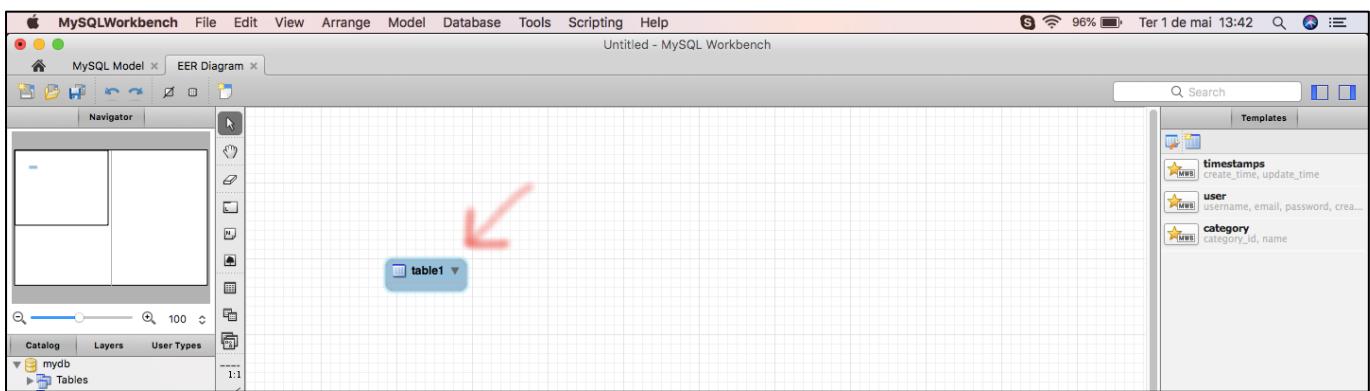


Figura 24 – Criando um novo schema, parte 4

Com o modo de edição da tabela aberto, troque o nome da tabela para “noticias” e adicione as seguintes colunas:

- Id\_noticia: tipo INT, PRIMARY KEY, NOT NULL, AUTO INCREMENT
- editoria: tipo VARCHAR(30), NOT NULL
- titulo: tipo VARCHAR(60), NOT NULL
- conteudo: tipo VARCHAR(2000), NOT NULL
- extensao\_imagem: tipo VARCHAR(5), NOT NULL
- data\_hora: tipo DATETIME, NOT NULL

Veja na figura 25 como a tabela deve ficar:

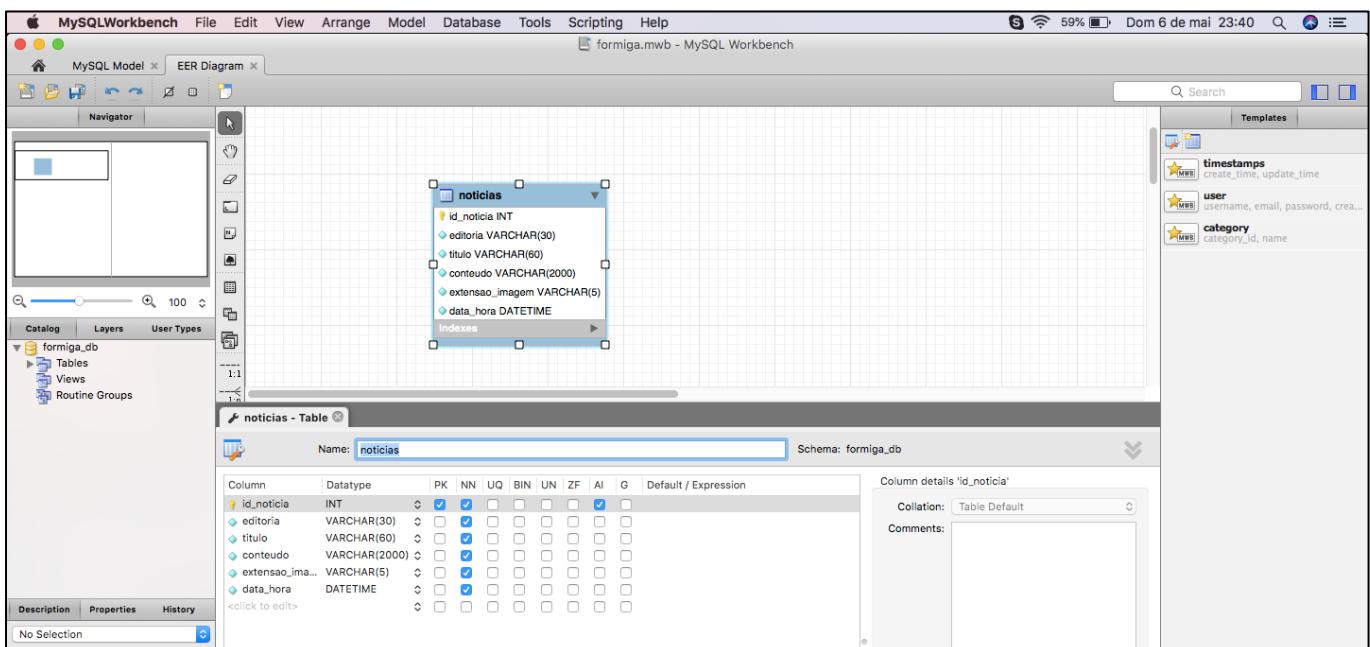


Figura 25 – Criando um novo schema, parte 5

## 5.6. Sincronizando o novo *schema* com o banco de dados

Agora que o *schema* está criado (não esqueça de salvá-lo), temos que sincronizá-lo com o banco de dados. Inicialmente, faremos a sincronização com o banco local e, em seguida, com o banco de produção. Na primeira sincronização, quando não existe nenhuma tabela criada no banco ainda, precisamos utilizar uma ferramenta chamada “*Forward Engineer...*”. Após essa sincronização inicial, passaremos a utilizar a ferramenta “*Synchronize Model...*” para atualizar as tabelas do banco. Isso serve tanto para o banco local, de desenvolvimento, quanto para o banco de produção.

Clique em “*Database*”, no menu superior, em seguida, em “*Forward Engineer...*”, conforme figura 26:

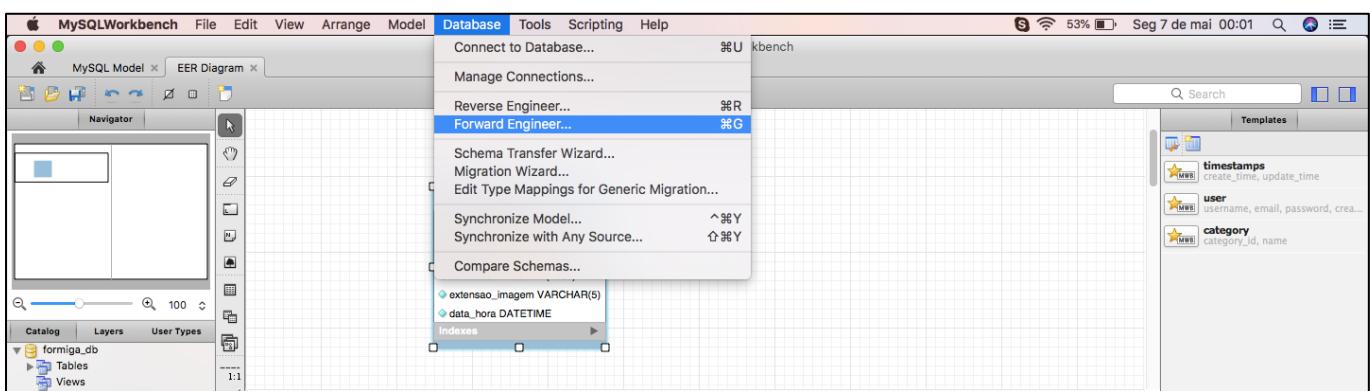


Figura 26 – Sincronizando o *schema* com o banco

Inicialmente, faça a sincronização com o banco local, selecionando localhost no campo “*Stored Connection*”. Em seguida, faça a sincronização com o banco de produção. Vá avançando as telas, confira o *script* que será executado e confirme a operação. A figura 27 mostra como deve ficar o *script*:

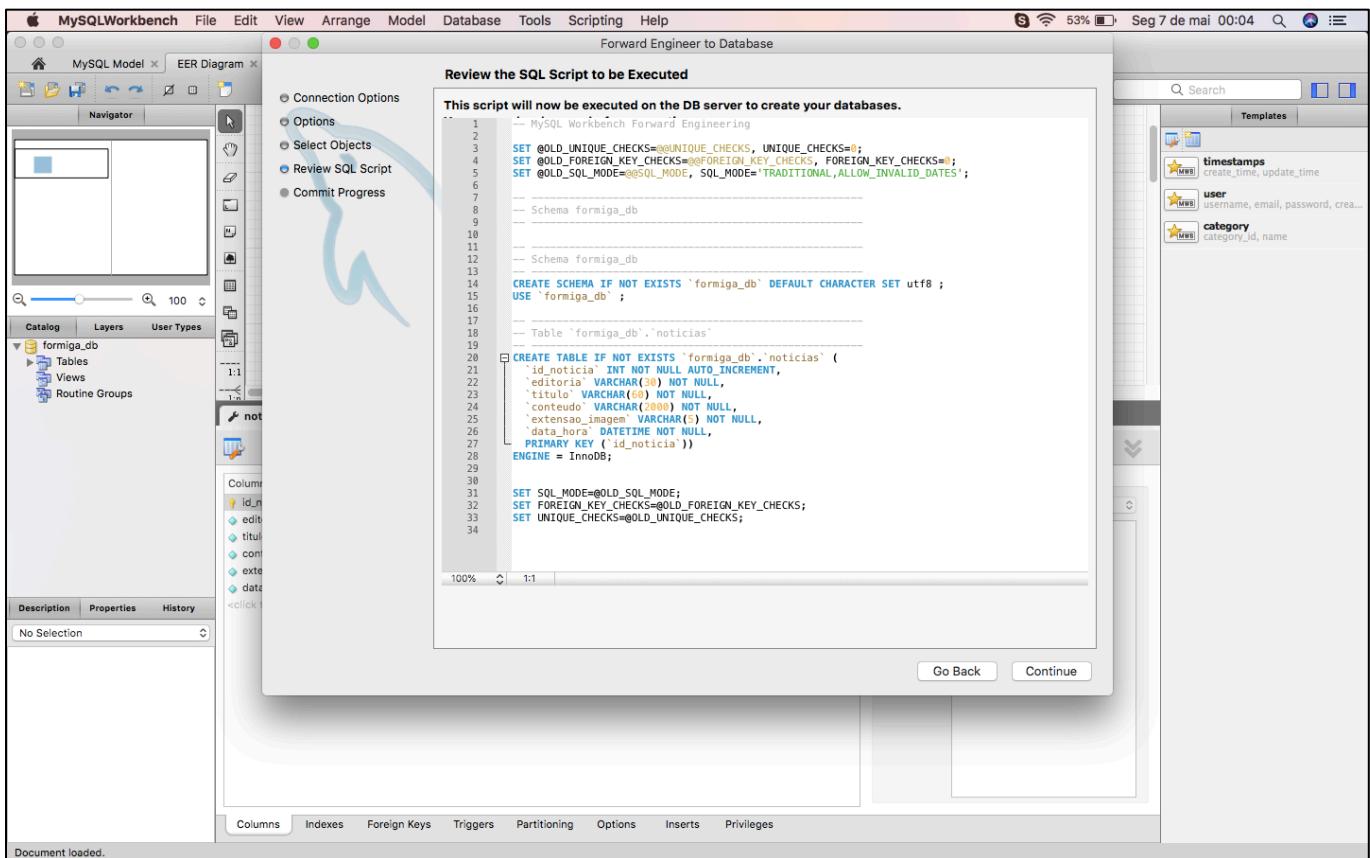


Figura 27 – Revisando o script de sincronização

Feita a sincronização, volte para a tela inicial do workbench, clique em conexões (ícone do golfinho) e, em seguida, clique na conexão localhost para conferir se a sincronização aconteceu corretamente. Deve ser possível visualizar a nova tabela “noticias” criada no banco de dados “formiga\_db”, conforme figura 28:

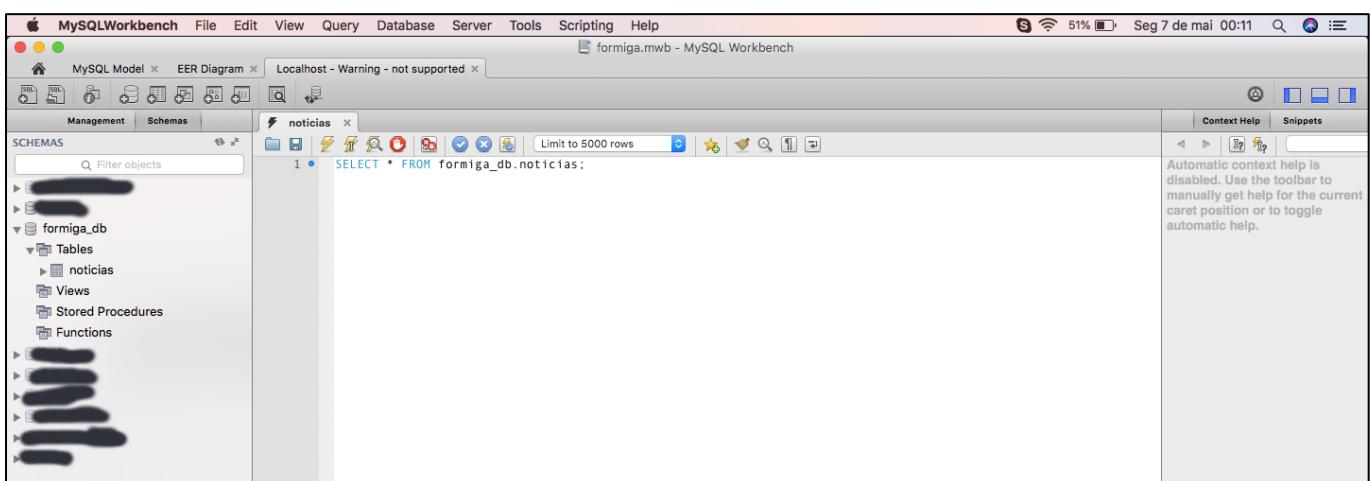


Figura 28 – Schema sincronizado com sucesso

Pronto, já é possível acessar e interagir com o banco através do código!

## 5.7. Código, parte 0: Entendendo a arquitetura

Agora que já temos o XAMPP instalado e funcionando e o banco de dados já sincronizado com o *schema*, podemos começar a desenvolver o código. O *backend*, que será feito em PHP, irá rodar no servidor e interagir diretamente com o banco de dados. A interface administrativa, que será feita “em web” (HTML + CSS + JS), irá rodar no browser e se comunicar com o *backend* através de requisições GET e POST que enviarão mensagens no formato JSON. Por fim, o *app*, que também será desenvolvido “em web”, será compilado e transformado em aplicativos instaláveis para rodar nos *devices* iOS/Android e se comunicar através de requisições GET e POST com o servidor, também enviando mensagens no formato JSON. À princípio, tudo será desenvolvido e testado localmente, com o auxílio do XAMPP e do Google Chrome. Em seguida, o *backend* e a interface administrativa vão para o servidor e o *app* será compilado.

Vale observar que, apesar de aqui o código estar dividido e de ser apresentado em diferentes seções, ele geralmente não é desenvolvido de forma tão separada quando se trata de um MVP. Isso acontece porque, dificilmente, uma parte (*frontend web*, *backend* e/ou *frontend app*) funciona por completo sem alguma das outras. Geralmente, o que acontece é um desenvolvimento em pequenos ciclos, no qual desenvolve-se alguma funcionalidade por completo (*front + back*), testa-se, e então passa-se para a próxima funcionalidade. Porém, aqui o resultado final será apresentado de uma vez para facilitar a compreensão.

## 5.8. Código, parte 1: *Backend* em PHP

Para este MVP, criaremos um *backend* em PHP bastante simples, porém com uma certa organização para facilitar qualquer desenvolvimento ou manutenção futura. Não será seguido nenhum padrão de mercado em termos de organização, e é válido dizer que não existe uma escolha correta. Existem muitas possibilidades de organização e muitas variáveis que devem ser levadas em consideração. Muitas vezes, uma boa escolha para um projeto não é uma boa escolha para outro. Como não faz parte dos objetivos deste guia entrar no mérito de padrões de organização de código e arquitetura, optaremos por criar a nossa própria. Isso, inclusive, ajuda a desmistificar a necessidade por escolher o melhor framework e facilita ao estudante aprender o

conceito que está por trás do funcionamento do código. Dito isso, vamos à nossa organização:

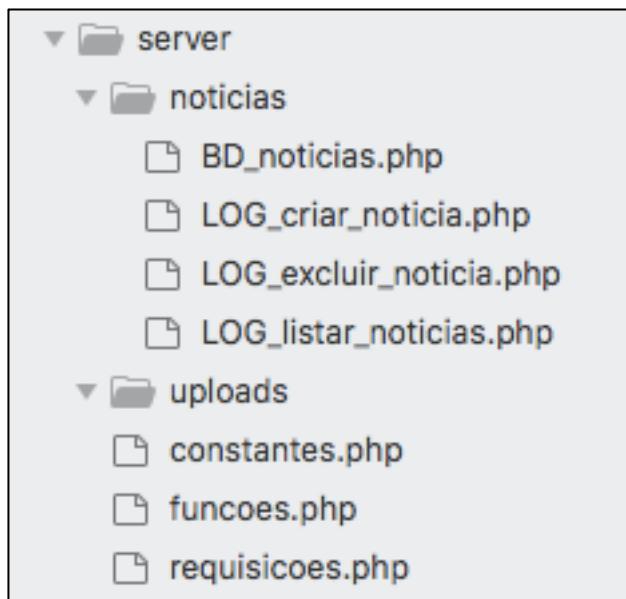


Figura 29 – Hierarquia de pastas e arquivos do backend PHP

Essa pasta “server” fica dentro da pasta raiz do projeto, “formiga”, que fica na pasta “htdocs” do nosso XAMPP para que o servidor rode corretamente localmente. É ela que contém todo o código do *backend* e ela deverá subir para o servidor de produção posteriormente. De modo geral, existem alguns arquivos comuns à qualquer serviço requisitado, arquivos que fazem parte de uma camada de lógica de negócio (iniciando com “LOG”) e arquivos que fazem parte de uma camada de interação com o banco de dados (iniciando com “BD”). A seguir, apresentam-se breves descrições de cada arquivo e seus respectivos conteúdos:

**5.8.1 constantes.php:** um arquivo que contém algumas constantes que podem ser utilizadas em vários pontos do código. A intenção de separar essas constantes num arquivo é facilitar o entendimento e qualquer alteração. Um bom, e recorrente, exemplo seria um caminho ou URL. Ao invés de escrevê-lo várias vezes ao longo do código e ter várias linhas para alterar, caso a URL mude, crie-se uma constante no arquivo de constantes e altere-se apenas essa linha. Outro bom exemplo é a facilidade em alterar as variáveis que fazem a conexão com o banco de dados local e de produção.

Basta comentar o conjunto de variáveis que não está sendo utilizado, de acordo com o ambiente. Para projetos maiores, recomenda-se utilizar variáveis de ambiente para essa função de forma a garantir maior segurança e fluidez no desenvolvimento e operação.

```
<?php

/**
 * Arquivo com constantes
 **/

// Dev
define('CONST_SERVER', 'localhost');
define('CONST_DB_USERNAME', 'root');
define('CONST_DB_PASSWORD', '');
define('CONST_DB_NAME', 'formiga_db');

// Prod
// define('CONST_SERVER', 'hostname_do_seu_servidor_mysql');
// define('CONST_DB_USERNAME', 'seu_username_no_banco_de_dados');
// define('CONST_DB_PASSWORD', 'suaSenha_no_banco_de_dados');
// define('CONST_DB_NAME', 'nome_do_seu_banco_de_dados');

// Imagens
define('CONST_PASTA_IMAGENS', $_SERVER['DOCUMENT_ROOT'] .
'/formiga/server/uploads');
```

**5.8.2 funções.php:** um arquivo que contém funções comuns, que podem ser usadas em vários pontos do código. Segue a mesma lógica do arquivo de constantes.

```
<?php

/**
 * Arquivo com funcoes auxiliares
 **/

/**
 * Ajusta data para horario de Sao Paulo
 *
 * @return date Data ajustada para horario de Sao Paulo
 */
function data_hora_saopaulo() {
    date_default_timezone_set('America/Sao_Paulo');
    $date = date('Y-m-d H:i:s');
    return $date;
}

/**
 * Extrai vetor de resultados de uma query no banco de dados
 */
```

```

/*
 *  @return array  Vetor com resultados de uma query no banco de
dados
*/
function extrai_array($result) {
    $array = array();
    while($row = $result->fetch_assoc()) {
        $array[] = $row;
    }
    return $array;
}

```

**5.8.3 requisicoes.php:** este arquivo será responsável por receber todas as requisições no *backend*, obter e “sanitizar” as variáveis, realizar a conexão com o banco de dados, fazer algumas verificações, caso haja algum arquivo, e direcionar a requisição para o serviço (função) correto.

```

<?php

/**
 *  Roteirizador - Recebe todas as requisicoes, sanitiza e pega os
inputs de GET e POST,
 *          faz verificacoes no caso de haver arquivos e
envia as informacoes
 *          para o servico correto
**/

// Seta headers
header("access-control-allow-origin: *");
header('Access-Control-Allow-Methods: GET, POST');

// Sanitiza os inputs externos (GET e POST)
$_GET    = filter_input_array(INPUT_GET, FILTER_SANITIZE_STRING);
$_POST   = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

// Inclui arquivo de constantes e funcoes comuns criado por nos
include_once('constantes.php');
include_once('funcoes.php');

// Cria conexao com o banco de dados
$con = new mysqli(CONST_SERVER, CONST_DB_USERNAME,
CONST_DB_PASSWORD, CONST_DB_NAME);
if($con->connect_error){
    die('Unable to connect to database [' . $con->connect_error .
']);
}

// Pega as variaveis de GET, se existir alguma
if(!empty($_GET)){
    foreach ($_GET as $key => $value) {
        $$key = $value;
    }
}

```

```

// Pega as variaveis de POST, se existir alguma
if(!empty($_POST)){
    foreach ($_POST as $key => $value) {
        $$key = $value;
    }
}

// Se existir algum arquivo na requisicao
if(isset($_FILES['file'])){

    $pasta = "uploads/";
    $arquivo = $pasta . basename($_FILES["file"]["name"]);
    $tipo_de_arquivo =
    strtolower(pathinfo($arquivo, PATHINFO_EXTENSION));

    // Verifica se o arquivo possui algum erro
    if($_FILES['file']['error'])
        exit(json_encode([
            'status' => 'Arquivo com erro: ' . $_FILES['file']['error']
        ]));

    // Verifica se o arquivo eh uma imagem real
    if(!getimagesize($_FILES["file"]["tmp_name"]))
        exit(json_encode([
            'status' => 'Este arquivo nao é uma imagem!'
        ]));

    // Verifica o tamanho do arquivo
    if($_FILES['file']['size'] > (300000))
        exit(json_encode([
            'status' => 'Arquivo muito grande! O limite é de 300kb.'
        ]));

    // Verifica se o arquivo possui uma extensao permitida
    if($tipo_de_arquivo != "jpg" && $tipo_de_arquivo != "png" &&
    $tipo_de_arquivo != "jpeg" && $tipo_de_arquivo != "gif" )
        exit(json_encode([
            'status' => 'Arquivo não permitido. Use a extensão jpg,
png, jpeg ou gif'
        ]));

    // Se nao existir um arquivo
} else {
    $tipo_de_arquivo = '';
}

// Direcciona a requisicao para o servico correto
switch ($func) {

    // Funcoes para noticias

    case 'criar_noticia':
        include_once('noticias/BD_noticias.php');
        include_once('noticias/LOG_criar_noticia.php');
        criar_noticia($titulo, $conteudo, $editoria, $tipo_de_arquivo);
        break;
}

```

```

case 'listar_noticias':
    include_once('noticias/BD_noticias.php');
    include_once('noticias/LOG_listar_noticias.php');
    listar_noticias();
    break;

case 'excluir_noticia':
    include_once('noticias/BD_noticias.php');
    include_once('noticias/LOG_excluir_noticia.php');
    excluir_noticia($id_noticia, $extensao_imagem);
    break;

// Default (caso o serviço chamado não exista)

default:
    echo json_encode(['status' => 'erro']);
    break;
}

```

**5.8.4 Pasta uploads:** esta pasta servirá para armazenar as imagens das notícias.

**5.8.5 Pasta noticias:** esta pasta guarda o código relacionado à lógica e interação com o banco de dados para os serviços relacionados a notícias.

Como nosso MVP é bastante enxuto, não existem pastas “irmãs” dela. Mas, ela serve de exemplo para organização de novas funcionalidades, como por exemplo, adicionar editorias às notícias. Neste caso, teríamos uma pasta chamada “editorias”, que armazenaria o código relacionado à lógica e interação com o banco de dados para funções relacionadas ao tema editoria.

**5.8.5.1 BD\_noticias.php:** este arquivo contém todas as funções relacionadas ao tema notícias que interagem com o banco de dados. Esta separação é interessante para que não haja linguagem SQL misturada com linguagem PHP nos arquivos que contém lógica de negócio. Sempre que alguma função da lógica de negócios precisar acessar o banco, ela deverá chamar alguma função deste arquivo, que receberá qualquer input necessário, interagirá com o banco e devolverá uma resposta.

```

<?php

/**
 * Camada de acesso aos dados - CRUD para Notícias

```

```

/**/

/**
*  CREATE
**/

/**
*    Cria uma nova noticia
*
*    @param string $titulo           Titulo da noticia
*    @param string $conteudo         Conteudo da noticia
*    @param string $editoria        Editoria na qual a noticia
esta inserida
*    @param string $tipo_de_arquivo Extensao do arquivo de
imagem
*    @param date   $data_hora       Data e hora da criacao da
noticia
*
*    @return int      ID da insercao no banco de dados, que eh o
mesmo usado para a noticia
**/
function BD_criar_noticia($titulo, $conteudo, $editoria,
$tipo_de_arquivo, $data_hora){
    $sql = "INSERT INTO noticias (editoria, titulo, conteudo,
extensao_imagem, data_hora)
VALUES ('$editoria', '$titulo', '$conteudo',
'$tipo_de_arquivo', '$data_hora') ";
    $result = $GLOBALS['con']->query($sql);
    return $GLOBALS['con']->insert_id;
}

/**/
*  READ
**/

/**
*    Lista todas as noticias cadastradas
*
*    @return array   Vetor com todas as propriedades de todas as
noticias cadastradas
**/
function BD_listar_noticias(){
    $sql = "SELECT * FROM noticias ";
    $result = $GLOBALS['con']->query($sql);
    $result = extrai_array($result);
    return $result;
}

/**/
*  DELETE
**/


/**

```

```

*     Deleta uma noticia especifica
*
*     @param int    $id_noticia    ID da noticia a ser deletada no
banco de dados
*
*     @return int    Quantidade de linhas afetadas pela delecao
*/
function BD_excluir_noticia($id_noticia){
    $sql = "DELETE FROM noticias WHERE id_noticia='$id_noticia'";
    $result = $GLOBALS['con']->query($sql);
    return $GLOBALS['con']->affected_rows;
}

```

**5.8.5.2 LOG\_criar\_noticia.php:** este arquivo contém a lógica de negócio necessária para criar uma notícia. Ele é chamado pelo arquivo requisicoes.php, recebe os inputs necessários, executa a lógica de negócio e dá uma resposta ao usuário.

```

<?php

/**
*     Camada logica - Cria uma nova noticia
*
*     @param string  $titulo          Titulo da noticia
*     @param string  $conteudo        Conteudo da noticia
*     @param string  $editoria       Editoria na qual a noticia
esta inserida
*     @param string  $tipo_de_arquivo Extensao do arquivo de
imagem
*
*     @return json      json contendo o status (ok ou erro) da operacao
*/
function criar_noticia($titulo, $conteudo, $editoria,
$tipo_de_arquivo){

    // Pega horario de Sao Paulo
    $data_hora = data_hora_saopaulo();

    // Cria uma nova noticia
    $id_noticia = BD_criar_noticia($titulo, $conteudo, $editoria,
$tipo_de_arquivo, $data_hora);

    // Se a noticia for criada corretamente
    if($id_noticia){

        // Se existe algum arquivo no upload
        if(isset($_FILES['file'])){

            // Monta nome real do arquivo com base no seu nome temporario
            // e na extensao
            $pasta = "uploads/";
            $arquivo = $pasta . basename($_FILES["file"]["name"]);

```

```

$tipo_de_arquivo =
strtolower(pathinfo($arquivo, PATHINFO_EXTENSION));
$arquivo = $pasta . $id_noticia . '.' . $tipo_de_arquivo;

// Move o arquivo para a pasta de uploads e verifica se esta
ok
if(!move_uploaded_file($_FILES['file']['tmp_name'],
$arquivo))
    exit(json_encode(['status' => 'erro']));
}

echo json_encode(['status' => 'ok']);

// Se houver problema na criacao da noticia
} else {
    echo json_encode(['status' => 'erro']);
}
}
}

```

**5.8.5.3 LOG\_listar\_noticias.php:** este arquivo contém a lógica de negócios necessária para se listar as notícias existentes e funciona de modo similar ao anterior.

```

<?php

/**
 *      Camada logica - Lista todas as noticias cadastradas
 *
 *      @return json      json contendo uma lista de todas as noticias
 cadastradas
 **/


function listar_noticias(){

// Pega uma lista com todas as noticias cadastradas
$result = BD_listar_noticias();

// Retorna json com a lista de noticias cadastradas
echo json_encode([
    'status' => 'ok',
    'noticias' => $result
]);
}

```

**5.8.5.4 LOG\_excluir\_noticia.php:** este arquivo contém a lógica de negócios necessária para se excluir uma notícia e funciona de modo similar ao anterior.

```

<?php

/**
 *      Camada logica - Exclui uma certa noticia
 *
 *      @param int      $id_noticia          ID da noticia a ser excluida
 *      no banco de dados
 *      @param string   $extensao_imagem   Extensao do arquivo a ser
 *      excluido na pasta de uploads
 *
 *      @return json    json contendo o status (ok ou erro) da operacao
 **/


function excluir_noticia($id_noticia, $extensao_imagem) {

    // Exclui uma noticia do banco
    $result = BD_excluir_noticia($id_noticia);

    // Se a noticia for excluida corretamente
    if($result){

        // Caso a noticia tenha uma imagem, deleta a imagem
        if($extensao_imagem != ''){
            $imagem = CONST_PASTA_IMAGENS . $id_noticia . '.' .
$extensao_imagem;
            unlink($imagem);
        }

        echo json_encode(['status' => 'ok']);

        // Se houver problema na exclusao da noticia
    } else {
        echo json_encode(['status' => 'erro']);
    }
}

```

Os arquivos, lógicas de negócio e interações com o banco de dados desenvolvidos nesse MVP são bastante simples, por isso pode parecer que essa organização é até exagerada. Porém, vale frisar que com ela é possível crescer bastante o MVP de forma organizada. Fica a cargo do estudante extrapolar isso para projetos mais complexos. As explicações mais detalhadas sobre o que cada arquivo faz se encontram nos comentários, no próprio código.

### 5.9. Código, parte 2: painel administrativo “em web”

Seguindo a mesma ideia do *backend*, a interface administrativa está bastante simples, mas também possui uma certa organização para facilitar o entendimento e para possibilitar crescimento futuro. Segue sua estrutura:

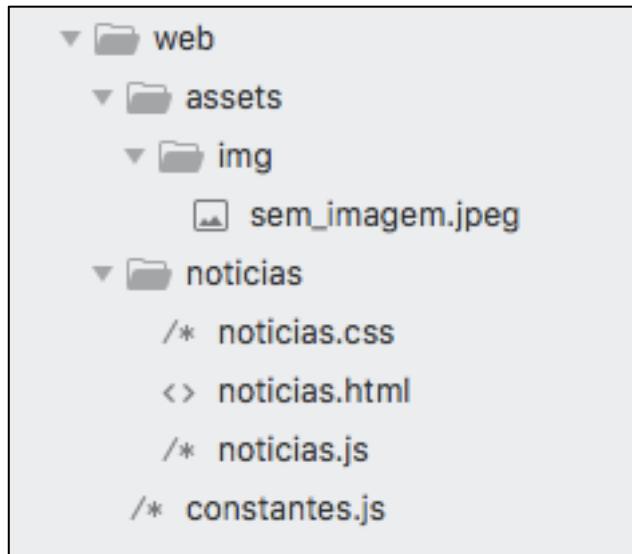


Figura 30 – Hierarquia de pastas e arquivos da interface administrativa

Essa pasta “web” também fica dentro da pasta “htdocs” do XAMPP, dentro da pasta “formiga”, do nosso projeto, no desenvolvimento local e, posteriormente, subirá para o servidor de produção. De modo geral, sua organização consiste em um arquivo com constantes, uma pasta com “ativos” (imagens apenas, no caso) e uma pasta que contém a interface administrativa em si, feita a partir de HTML, CSS e JS. A seguir, uma breve descrição de cada arquivo, seguida de seu conteúdo:

**5.9.1 constantes.js:** seguindo a mesma lógica do arquivo de constantes do PHP, este arquivo serve para centralizar as diversas constantes e facilitar alterações no projeto, uma vez que torna possível utilizar as constantes ao longo do código e manter suas definições num único lugar.

```

/**
 *   Arquivo com constantes
 */

/**
 *   Ambiente - Comentar o que nao esta sendo usado
 */
// var ambiente = 'http://localhost/formiga/' // Dev
var ambiente = 'http://www.formiga.info/' // Prod

/**
 *   URLs para as requisicoes
 */
var host = ambiente + 'server/requisicoes.php'
var imagens = ambiente + 'server/uploads/'

/**

```

```
*   Nomes dos serviços para notícias
*/
var func_criar_noticia = 'criar_noticia'
var func_listar_noticias = 'listar_noticias'
var func_excluir_noticia = 'excluir_noticia'
```

5.9.2 **Pasta assets**: esta pasta contém os “ativos” do projeto, no caso, arquivos de imagem apenas.

5.9.2.1 **Pasta img**: pasta que contém as imagens do projeto.

5.9.2.1.1 **sem\_imagem.jpeg**: imagem que aparece nas notícias que não possuem imagem.

5.9.3 **Pasta notícias**: esta pasta contém a interface administrativa para publicação de notícias

5.9.4 **noticias.html**: este é o arquivo html responsável pela estrutura da página de administração. Ele é bastante simples e limpo, faz uso do bootstrap 4 para facilitar sua estruturação e responsividade e consiste, basicamente, em uma área para cadastro de novas notícias, que fica na coluna da esquerda em dispositivos grandes, e um área para visualização e deleção das notícias cadastradas, que fica na coluna da direita em dispositivos grandes. Em dispositivos pequenos, como celulares, essas áreas ficam empilhadas. Não vamos entrar em detalhes sobre o funcionamento do bootstrap, pois ele possui uma ótima documentação disponível online no mesmo endereço apresentado na seção 3.14. Esta documentação também possui um exemplo de arquivo html 5 para ser utilizado juntamente com o bootstrap. Também não vamos entrar em detalhes sobre a estrutura de um arquivo html para não estender nem perder o foco do guia. Sobre o código em si, é válido destacar que no <head> do html estamos incluindo o arquivo de estilos que criamos, o noticias.css, a partir de um endereço relativo que aponta para a mesma pasta em que o html está, e o arquivo de estilos do bootstrap, que está apontando para a CDN indicado pelos próprios desenvolvedores do bootstrap. Uma CDN é uma *Content Distribution Network*, uma rede global de servidores que vai entregar arquivos mais próximos do ponto de uso. Sempre que um projeto rodar na web e utilizar alguma biblioteca externa, é

recomendável utilizar o CDN indicado pelo desenvolvedor, se houver. Isso traz vantagens em termos de velocidade de carregamento, pois dificilmente o seu servidor entregará o arquivo mais rapidamente que a CDN, e custos, pois essa carga de trabalho deixa de ser do seu servidor. No final do <body>, para não atrasar o carregamento do html, estão as inclusões dos arquivos javascript utilizados, dois criados por nós (constantes.js e noticias.js) e três necessários ao funcionamento do bootstrap, também buscados nas CDNs indicadas pelos desenvolvedores. Além dos arquivos externos, temos um pequeno trecho de javascript interno que é responsável por listar as notícias existentes, uma vez que o documento esteja completamente carregado, e por apresentar o botão de excluir notícia quando o mouse estiver em cima de uma.

```
<!doctype html>
<html lang="en">
<head>

    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <!-- Para usar na web, eh recomendado buscar o arquivo no CDN
indicado pelo desenvolvedor -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.
min.css" integrity="sha384-WskhaSGFgHYWDCbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB"
crossorigin="anonymous">
    <link rel="stylesheet" href="noticias.css">

    <title>Notícias</title>

</head>
<body>

    <div class="container">

        <div class="row">

            <!-- Formulario para cadsaturo de nova noticia -->
            <div class="col-12 col-lg-5">
                <br>
                <h1>Nova notícia</h1>
                <br>
                <form>
```

```

<div class="form-group">
    <label for="editoria">Editoria</label>
    <input type="text" class="form-control" id="editoria"
placeholder="Editoria" maxlength="30">
</div>
<div class="form-group">
    <label for="titulo">Título</label>
    <input type="text" class="form-control" id="titulo"
placeholder="Título da notícia" maxlength="60">
</div>
<div class="form-group">
    <label for="conteudo">Conteúdo</label>
    <textarea class="form-control" id="conteudo"
placeholder="Conteúdo da notícia" rows="6"
maxlength="2000"></textarea>
</div>
<div class="form-group">
    <label for="imagem">Imagem</label>
    <input type="file" class="form-control-file"
id="imagem" name="imagem">
</div>
<br>
<button type="button" class="btn btn-primary"
onclick="criar_noticia()">Criar notícia</button>
<br>
<br>
<div id="resp_noticia"></div>
</form>
</div>

<!-- Área para listagem das notícias cadastradas -->
<div class="col-12 col-lg-7">
    <br>
    <h1>Notícias</h1>
    <br>
    <div id="noticias"></div>
</div>
</div>

</div>

<!-- JS necessários para o bootstrap 4.1 (jQuery completo e não
slim) -->
<!-- Para usar na web, é recomendado buscar o arquivo no CDN
indicado pelo desenvolvedor -->
<script src="https://code.jquery.com/jquery-3.3.1.min.js"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/po
pper.min.js" integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLauAdn689aCwoqbBJiSnjAK/18WvCWPIPM49"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap
.min.js" integrity="sha384-
smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+5T"
crossorigin="anonymous"></script>

```

```

<!-- JS externo ao html -->
<script type="text/javascript" src="../constantes.js"></script>
<script type="text/javascript" src="noticias.js"></script>

<!-- JS interno ao html -->
<script type="text/javascript" charset="UTF-8">

$(document).ready(function(){

    // Lista noticias
    listar_noticias()

    // Faz botao de excluir noticia aparecer quando o mouse esta
    em cima dela e sumir quando nao esta
    $(document).on('mouseenter', '.div-noticia', function () {
        $(this).find(".excluir-noticia").show()
    }).on('mouseleave', '.div-noticia', function () {
        $(this).find(".excluir-noticia").hide()
    });
})

</script>

</body>
</html>

```

**5.9.5 *noticias.css*:** este é o arquivo de estilos, responsável por dar uma cara personalizada para nosso html. O bootstrap já possui alguns estilos pré-definidos bastante úteis, mas quando deseja-se customizar ainda mais o html, criam-se estilos próprios. Nas classes dos elementos html é possível ver quais são os estilos utilizados e fica fácil distinguir quais são nativos do bootstrap, pois estes não fazem parte deste arquivo.

```

.excluir-noticia{
    position:absolute;
    left:80%;
    display:none;
}

.editoria_noticia{
    color:gray;
}

.cor_resp_positiva{
    color:green;
}

.cor_resp_negativa{
    color:red;
}

```

5.9.6 **noticias.js**: este é o arquivo responsável pelas funções que comunicam com o servidor: criar noticia, listar noticias e deletar noticia. Também não entraremos em detalhes de como o javascript funciona. Assim como os demais, este arquivo possui comentários que visam deixar mais claro o que cada parte do código faz.

```
/**
 *   Lista noticias
 */
function listar_noticias(){

    // Realiza um GET para buscar as informacoes do servidor
    $.get(host, {func: func_listar_noticias}, function(data){

        // Le a resposta em formato JSON
        var result = JSON && JSON.parse(data) || $.parseJSON(data);

        // Limpa html para receber nova lista de noticias
        $('#noticias').empty()

        // Se houver alguma noticia cadastrada, gera lista de noticias
        if(result.noticias.length > 0){
            var noticias = ''
            for(i = 0; i < result.noticias.length; i++){
                noticias +=
                    '<div class="row div-noticia">'+
                    '<div class="col-3">'

                    // Verifica se a noticia possui imagem
                    if(result.noticias[i].extensao_imagem == ''){
                        noticias += ''
                    } else
                        noticias += ''

                noticias +=
                    '</div>'+
                    '<div class="col-9">'+
                    '<button class="btn btn-danger excluir-noticia"'+
                    'onclick="excluir_noticia('+result.noticias[i].id_noticia+', \''+resu+
                    lt.noticias[i].extensao_imagem+'\')">Excluir</button>'+
                    '<h4>' + result.noticias[i].titulo + '</h4>'+
                    '<p'

                class="editoria_noticia">' + result.noticias[i].editoria +
                (' + result.noticias[i].data_hora.substring(8,10) + '/' + result.noticias
                [i].data_hora.substring(5,7) + ' - 
                ' + result.noticias[i].data_hora.substring(11,13) + ':' + result.noticias
                [i].data_hora.substring(14,16) + ') </p>'+
                    '<p>' + result.noticias[i].conteudo + '</p>'+
                    '</div>'+
```

```

        '</div>'+
        '<hr>'+
    }
    $('#noticias').append(noticias)

    // Se nao houver nenhuma noticia cadastrada
} else {
    $('#noticias').append('<br>Não há notícias
cadastradas<br><br>')
}
}

/***
 *   Cria nova noticia
***/
function criar_noticia(){

    // Pega valores do formulario
    var editoria = $('#editoria').val()
    var titulo = $('#titulo').val()
    var conteudo = $('#conteudo').val()

    // Verifica se existe uma editoria
    if(editoria == ''){
        alert('Escolha uma editoria')
        return false

    // Verifica se existe um titulo
    } else if(titulo == ''){
        alert('Dê um título para a notícia')
        return false

    // Verifica se existe conteudo
    } else if(conteudo == ''){
        alert('Escreva um conteúdo para a notícia')
        return false
    }

    // Monta formulario para ser enviado (este POST eh diferente,
    // pois pode ter um arquivo, entao precisa do FormData)
    dataform = new FormData()
    dataform.append( 'file', $( '#imagem' )[0].files[0] )
    dataform.append( 'editoria', editoria )
    dataform.append( 'titulo', titulo )
    dataform.append( 'conteudo', conteudo )
    dataform.append( 'func', func_criar_noticia )

    // Faz o POST para o servidor e envia as informacoes
    $.ajax({
        url: host,
        data: dataform,
        cache: false,
        contentType: false,
        processData: false,
        method: 'POST',

```

```

// No caso de sucesso
success: function(data) {

    var result = JSON && JSON.parse(data) || $.parseJSON(data)

    if(result.status == 'ok'){

        // Limpa campos
        $('#editoria').val('')
        $('#titulo').val('')
        $('#conteudo').val('')
        var e = $('#imagem')
        e.wrap('<form>').closest('form').get(0).reset()
        e.unwrap()

        // Insere aviso temporario de status positivo
        $('#resp_noticia').append('<span
class="cor_resp_positiva">Notícia criada!</span>')
        setTimeout(function() { $('#resp_noticia').fadeOut(800); },
800)
        listar_noticias()

    } else {
        // Insere aviso temporario de status negativo
        $('#resp_noticia').append('<span
class="cor_resp_negativa">Ocorreu um erro!</span>')
        setTimeout(function() { $('#resp_noticia').fadeOut(800); },
800);
    }
}

// No caso de falha
error: function(data){
    alert('Ops, ocorreu um problema no upload.')
}
});

}

/***
 * Exclui noticia
 ***/
function excluir_noticia(id_noticia, extensao_imagem){

    // Faz um POST para enviar informacoes ao servidor
    $.post(host, {func: func_excluir_noticia, id_noticia: id_noticia,
extensao_imagem: extensao_imagem}, function(data){

        // Le a resposta em formato JSON
        var result = JSON && JSON.parse(data) || $.parseJSON(data);

        // Se a exclusao ocorreu com sucesso
        if(result.status == 'ok'){
            listar_noticias()

        // Se houve alguma falha na exclusao
        } else {
            alert('Não foi possível excluir a noticia!')
        }
    });
}

```

```
        }  
    })  
}
```

O mesmo comentário sobre a organização do código feito para o *backend* se aplica também para a interface administrativa. Não há escolha correta sobre organização, mas existem algumas boas práticas, como separar html, css e js em arquivos devidamente identificados quando não estamos trabalhando com frameworks que seguem a lógica de componentes.

Com a interface administrativa (pasta web) e o *backend* (pasta server) dentro de uma pasta raiz chamada “formiga”, na pasta htdocs do XAMPP e as demais configurações feitas, já é possível visualizar e interagir com o painel administrativo de gerenciamento das notícias através do browser, pelo endereço “localhost/formiga/web/noticias/noticias.html”, conforme imagem a seguir:

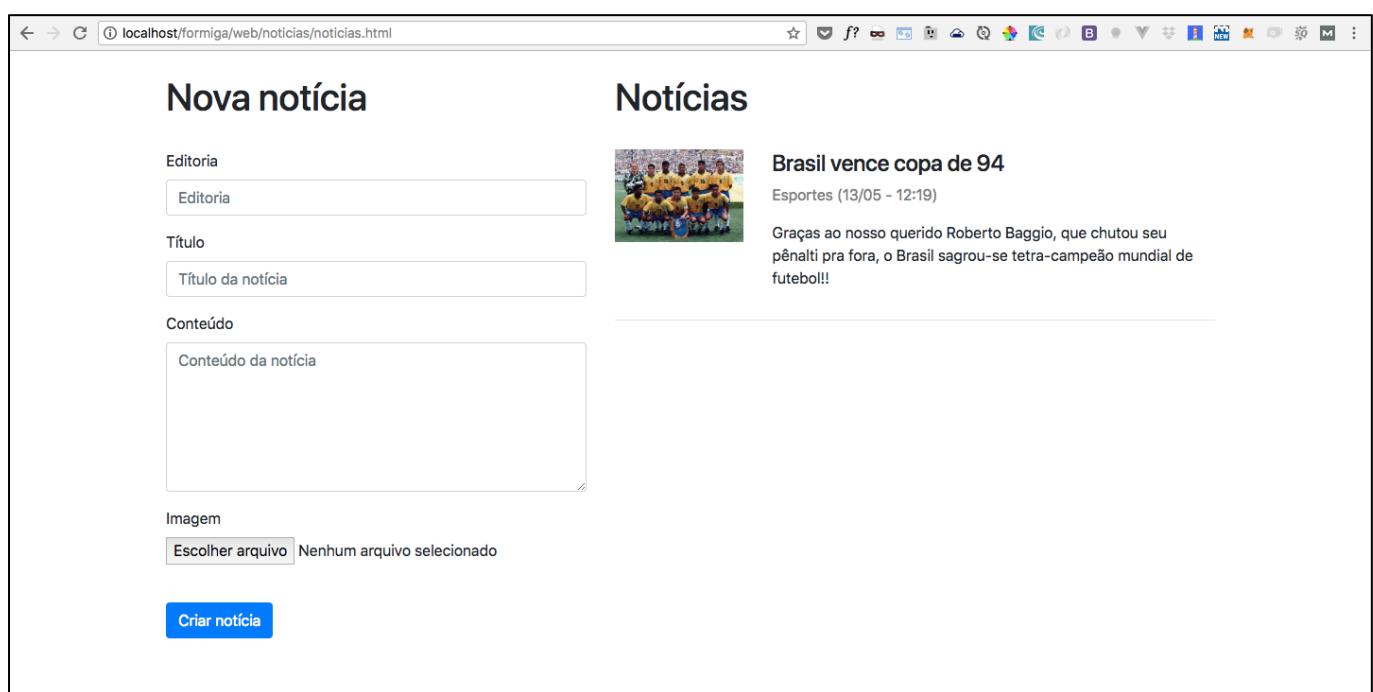


Figura 31 – Interface administrativa rodando em ambiente local

Atente-se para a estrutura de pastas, um engano comum no inicio do aprendizado é tentar acessar arquivos através de URLs erradas. Quando digitamos “localhost/” no browser, estamos indo para a pasta htdocs do XAMPP, portanto, atente-se para o caminho correto de seus arquivos. No nosso caso, temos uma pasta raiz do projeto

chamada “formiga” e dentro dela temos as pastas “web” e “server”, que mostramos anteriormente. Outros enganos comuns são esquecer de ativar o Apache ou o MySQL no painel administrativo do XAMPP, ou utilizar caminhos errados dentro do código, no nosso caso, nos arquivos de constantes.

Um bom aliado para identificar qualquer problema é o painel de desenvolvedor do Google Chrome, no qual é possível ver quais são as requisições que estão sendo feitas, se há algum erro de javascript, entre outros. Ele pode ser acessado pelos atalhos F12 ou ctrl + shift + i, no Windows, ou option + command + i, no mac.

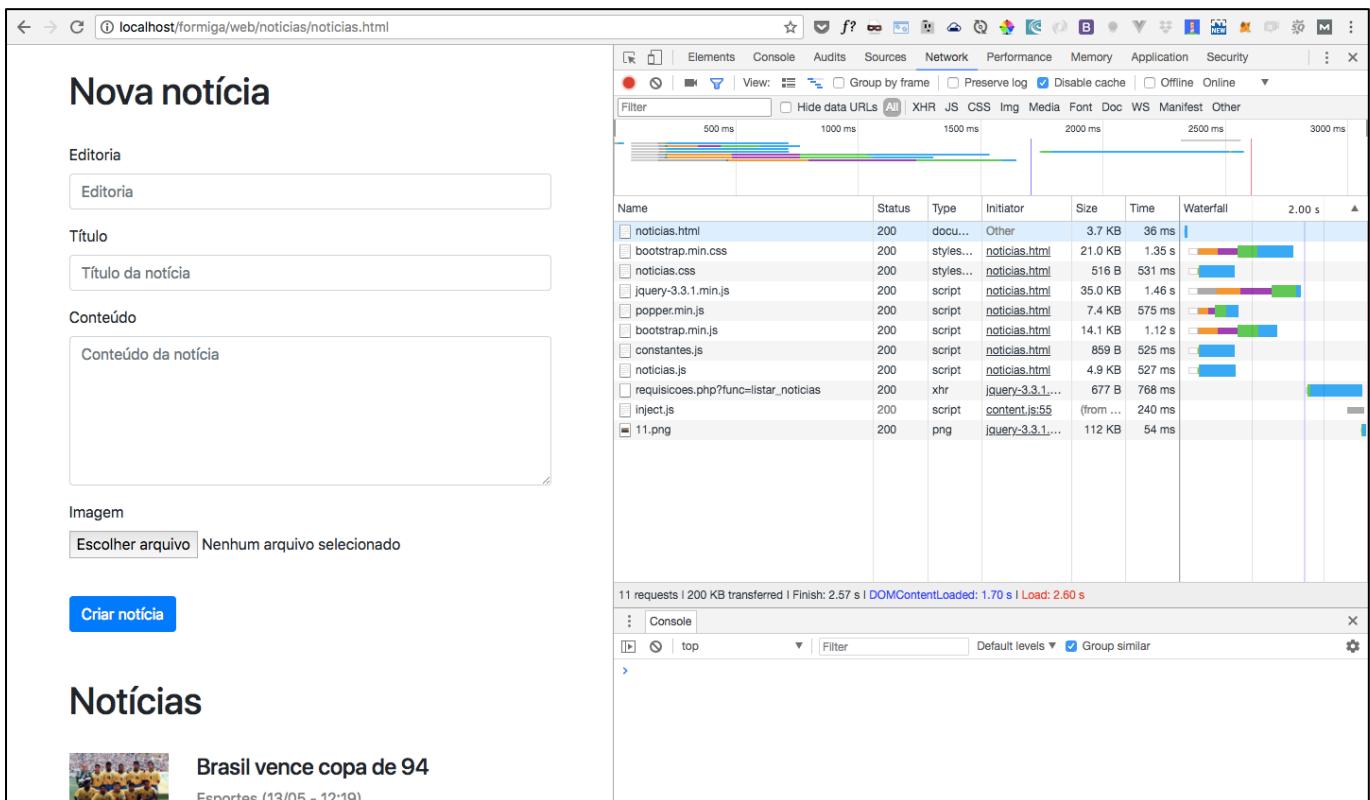


Figura 32 – Painel de desenvolvedor do Google Chrome

### 5.10 Código, parte 3: app “em web”

Agora que já temos o painel administrativo para gestão de notícias funcionando, falta apenas criar o *app* para que os usuários possam visualizar as notícias cadastradas. Mais uma vez, a estrutura do aplicativo está bastante simples, mas segue uma certa organização. Neste caso, como vamos usar a ferramenta PhoneGap Build para realizar a compilação do código e criação dos instaláveis, temos um arquivo XML de configurações e uma pasta “www” que contém o código para atender a requisitos de organização

necessários para que a ferramenta possa executar corretamente a compilação online. É importante que o arquivo “config.xml” fique no mesmo nível da pasta “www” e que na hora do upload, seja enviada a pasta mãe destas destacadas zipada. Essa pasta não precisa ter o nome “app”, ela pode ter qualquer nome. Ainda sobre a questão organização, por padrão, o aplicativo tentará abrir a página index.html inicialmente, dentro da pasta “www”. A organização do código do aplicativo fica da seguinte forma:

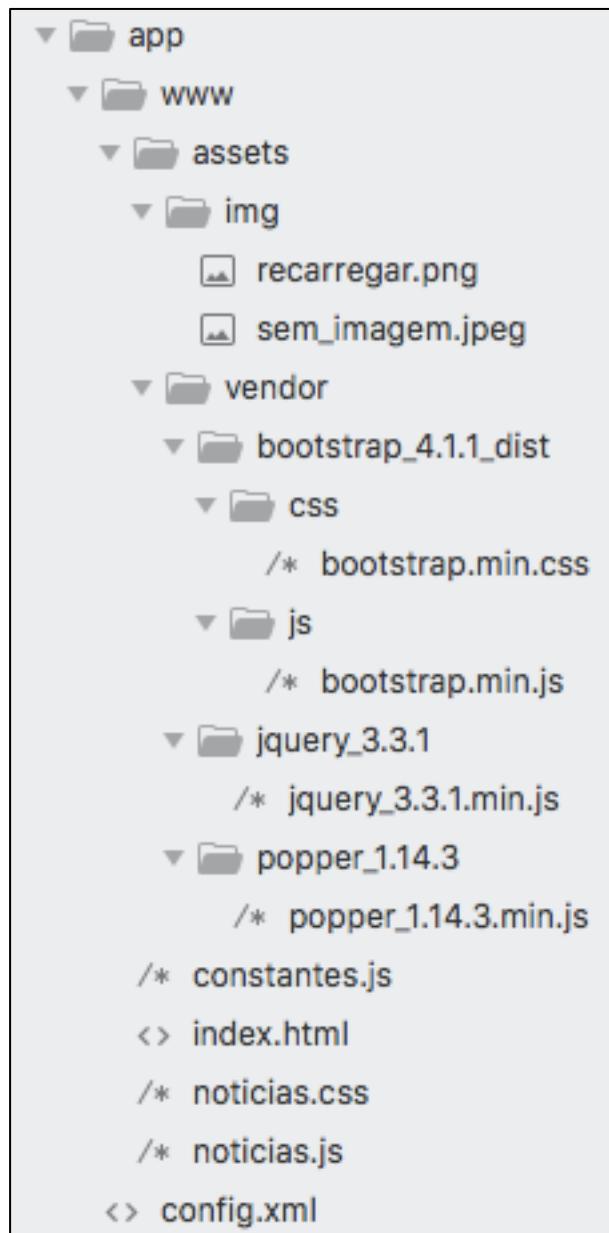


Figura 33 – Estrutura de pastas e arquivos do app

Ditos os pontos mais importantes sobre a organização, relacionados ao PhoneGap Build, passamos para uma breve descrição e apresentação do conteúdo dos arquivos que desenvolvemos:

#### 5.9.1. Arquivo config.xml

Este xml é um arquivo de configuração utilizado pelo PhoneGap e pelo PhoneGap Build para compilar o código web e transformá-lo em arquivos instaláveis para iOS e Android.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<widget xmlns      = "http://www.w3.org/ns/widgets"
         xmlns:gap    = "http://phonegap.com/ns/1.0"
         id          = "info.formiga.teste"
         versionCode = "1"
         version     = "0.1.0">

    <name>Formiga news</name>
    <description>Formiga: seu app de notícias</description>

    <!-- WHITE LIST-->
    <plugin name="cordova-plugin-whitelist" source="npm" />
    <access origin="*://www.formiga.info" subdomains="true" />

    <!-- Versao do PhoneGap -->
    <preference name="phonegap-version" value="cli-8.0.0" />

    <!-- Informacoes do desenvolvedor -->
    <author href="https://www.formiga.info"
            email="fabbri07@gmail.com">BRUNO FABBRI</author>

    <!-- Desabilitar overscroll -->
    <preference name="DisallowOverscroll" value="true" />

    <!-- Viewport vertical -->
    <preference name="orientation" value="portrait" />

    <!-- Escondendo o statusbar -->
    <edit-config target="UIStatusBarHidden" file="*-Info.plist"
                mode="overwrite" platform="ios">
        <true/>
    </edit-config>
    <edit-config target="UIViewControllerBasedStatusBarAppearance"
                file="*-Info.plist" mode="overwrite" platform="ios">
        <false/>
    </edit-config>

</widget>
```

### **5.9.2. Pasta www**

A pasta www, como dito anteriormente, contém o código do nosso aplicativo e segue a organização necessária para o PhoneGap Build “entender” nosso código.

### **5.9.3. Pasta www/assets**

Assim como na interface web, a pasta assets contém os “ativos” do nosso projeto.

#### **5.9.4. Pasta www/assets/img**

Assim como na interface web, a pasta img contém as imagens do nosso projeto.

##### **5.9.5. Arquivo www/assets/img/recarregar.png**

Esta imagem simboliza a ação de recarregar a página para buscar por novas notícias e é utilizada no canto superior direito do aplicativo.

##### **5.9.6. Arquivo www/assets/img/sem\_imagem.jpeg**

Esta imagem aparece para as notícias que não contém imagem.

#### **5.9.7. Pasta www/assets/vendor**

Esta pasta contém os ativos do projeto que foram desenvolvidos por terceiros.

#### **5.9.8. Pasta www/assets/vendor/bootstrap\_4.1.1\_dist**

Esta pasta contém os arquivos css e js utilizados pelo Bootstrap 4. No caso do aplicativo, como ele não precisa necessariamente carregar os arquivos da web, como é o caso da interface administrativa, é mais interessante empacotar todos os recursos necessários para seu funcionamento dentro dos instaláveis, assim melhora-se o tempo de carregamento e economizam-se requisições e banda de internet.

#### **5.9.9. Pasta www/assets/vendor/bootstrap\_4.1.1\_dist/css**

Esta pasta contém os arquivos css do bootstrap 4.

#### **5.9.10. Arquivo**

##### **www/assets/vendor/bootstrap\_4.1.1\_dist/css/bootstrap.min.css**

Este arquivo contém os estilos pré-definidos do bootstrap 4. Seu conteúdo não será apresentado, pois ele não foi desenvolvido pelo autor.

#### **5.9.11. Pasta www/assets/vendor/bootstrap\_4.1.1\_dist/js**

Esta pasta contém os arquivos js do bootstrap 4.

#### **5.9.12. Arquivo**

##### **www/assets/vendor/bootstrap\_4.1.1\_dist/js/bootstrap.min.js**

Este arquivo contém as funções em javascript que são utilizadas pelo bootstrap 4. Seu conteúdo não será apresentado, pois ele não foi desenvolvido pelo autor.

### **5.9.13. Pasta www/assets/vendor/jquery\_3.3.1**

Esta pasta contém o jquery, uma biblioteca javascript necessária para o funcionamento completo do bootstrap 4.

### **5.9.14. Arquivo www/assets/vendor/jquery\_3.3.1/jquery\_3.3.1.min.js**

Biblioteca javascript necessária para o funcionamento completo do bootstrap 4. Seu conteúdo não será apresentado, pois ela não foi desenvolvida pelo autor.

### **5.9.15. Pasta www/assets/vendor/popper\_1.14.3**

Esta pasta contém o popper, uma biblioteca javascript necessária para o funcionamento completo do bootstrap 4.

### **5.9.16. Arquivo www/assets/vendor/popper\_1.14.3/popper\_1.14.3.min.js**

Biblioteca javascript necessária para o funcionamento completo do bootstrap 4. Seu conteúdo não será apresentado, pois ela não foi desenvolvida pelo autor.

### **5.9.17. Arquivo www/constantes.js**

Este arquivo, assim como os arquivos de constantes da pasta server e da pasta web, contém algumas constantes que são utilizadas ao longo do código.

```
/**
 *   Arquivo com constantes
 **/

/**
 *   Ambiente - Comentar o que nao esta sendo usado
 **/
// var ambiente = 'http://localhost/formiga/' // Dev
var ambiente = 'http://www.formiga.info/' // Prod

/**
 *   URLs para as requisicoes
 **/
var host = ambiente + 'server/requisicoes.php'
var imagens = ambiente + 'server/uploads/'

/**
 *   Nomes dos servicos para noticias
 **/
var func_listar_noticias = 'listar_noticias'
```

Neste ponto, os estudantes mais atentos podem estar se perguntando porque as constantes estão sendo declaradas com “var” e não com “const”. Isso é proposital, pois a especificação de “const” surgiu apenas no ECMAScript 2015, ou ECMAScript 6, e essa

versão não é suportada por navegadores antigos. Portanto, principalmente nos arquivos javascript do aplicativo, deve-se tomar cuidado especial para não usar nenhuma sintaxe mais recente do que o ECMAScript 5, pois como o phonegap simula um browser nativo, qualquer sintaxe mais recente gerará um erro de javascript que travará a execução do código e, consequentemente, a aplicação em *devices* mais antigos, como aqueles com Android < 4.4. Com o passar do tempo, esses *devices* mais antigos se tornarão uma porcentagem irrelevante do mercado, tornando seguro o uso de especificações mais recentes do ECMAScript. Talvez, alguém também tenha notado que as constantes estão em caixa baixa, isso é porque o javascript aceita tanto caixa alta quanto caixa baixa na declaração de constantes.

#### 5.9.18. Arquivo www/index.html

Este arquivo html é bastante parecido com o que apresentamos na interface web. Ele possui uma área para exibição de notícias e uma barra superior com o nome do *app* e um botão para atualizar a lista de notícias. Ele também utiliza o bootstrap 4 para ajudar na responsividade e facilitar o desenvolvimento. A grande diferença aqui é que os arquivos css e js necessários ao funcionamento do bootstrap estão em pastas locais e são compilados junto com o código principal do app, ao invés de serem referenciados em CDNs. Fazemos isso para aumentar a velocidade de carregamento, diminuir a dependência de fontes externas e reduzir o uso de banda, que, muitas vezes, pode ser 3/4G. Assim como no html da interface administrativa, os arquivos de estilo também são chamados dentro do <head> e os arquivos javascript são chamados ao final do <body>. Há, ainda, um javascript interno que é responsável por buscar as notícias assim que o documento está completamente carregado. Uma outra particularidade deste html é que ele possui um modal que se abre com mais detalhes de uma notícia quando clicamos sobre ela.

```
<!doctype html>
<html lang="en">
<head>

    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
```

```

<!-- Para usar no app, eh recomendado embarcar o arquivo junto
com o codigo do app -->
<link rel="stylesheet"
href="assets/vendor/bootstrap_4.1.1_dist/css/bootstrap.min.css">
<link rel="stylesheet" href="noticias.css">

<title>Formiga News</title>

</head>
<body>

<div class="container">

<!-- Barra superior -->
<nav class="navbar fixed-top navbar-light bg-light">
    <a class="navbar-brand" href="#">Formiga News</a>
    <div class="float-right">
        
    </div>
</nav>

<!-- Area para as noticias -->
<div id="noticias"></div>

<!-- Modal para abertura das noticias -->
<div class="modal fade" id="modalNoticia" tabindex="-1"
role="dialog" aria-labelledby="tituloModalNoticia" aria-
hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">

                <!-- Aqui vai o titulo da noticia -->
                <h5 class="modal-title" id="tituloModalNoticia"></h5>

                <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>

                <!-- Aqui vai o conteudo da noticia -->
                <div class="modal-body" id="conteudoModalNoticia"></div>

            </div>
        </div>
    </div>
</div>

<!-- JS necessarios para o bootstrap 4.1 (jQuery completo e nao
slim) -->
<!-- Para usar no app, eh recomendado embarcar os arquivos junto
com o codigo do app -->
<script
src="assets/vendor/jquery_3.3.1/jquery_3.3.1.min.js"></script>
<script
src="assets/vendor/popper_1.14.3/popper_1.14.3.min.js"></script>

```

```

<script
src="assets/vendor/bootstrap_4.1.1_dist/js/bootstrap.min.js"></script>

<!-- JS externos ao html -->
<script src="constantes.js"></script>
<script src="noticias.js"></script>

<!-- JS interno ao html -->
<script type="text/javascript" charset="UTF-8">

$(document).ready(function(){

    // Lista noticias
    listar_noticias()

})

</script>

</body>
</html>

```

#### **5.9.19. Arquivo www/noticias.css**

Este arquivo contém os estilos criados por nós para o aplicativo.

```

body{
    padding-top: 72px;
}

.editoria_noticia{
    color:gray;
}

```

#### **5.9.20. Arquivo www/noticias.js**

Este arquivo contém a função que interage com o servidor para buscar pelas notícias recentes e a função que abre uma notícia com mais detalhes.

```

/**
 *   Lista noticias na tela
 */
function listar_noticias(){
    // Realiza um GET para buscar as informacoes do servidor
    $.get(host, {func: func_listar_noticias}, function(data){

        // Le a resposta em formato JSON
        var result = JSON && JSON.parse(data) || $.parseJSON(data);

        // Salva as noticias numa variavel local para serem acessadas
        // quando a noticia for aberta
        for(i = 0; i < result.noticias.length; i++) {

```

```

        localStorage.setItem("noticia-
"+result.noticias[i].id_noticia,
JSON.stringify(result.noticias[i]));
    }

    // Limpa html para receber nova lista de noticias
    $('#noticias').empty()

    // Se houver alguma noticia cadastrada, gera lista de noticias
    if(result.noticias.length > 0){

        // Cria e monta bloco de html
        var noticias = ''
        for(i = 0; i < result.noticias.length; i++){
            noticias +=
                '<div class="row" data-toggle="modal" data-
target="#modalNoticia"
onclick="abrir_noticia('+result.noticias[i].id_noticia+')">' +
                '<div class="col-3">'

                // Verifica se a noticia possui imagem
                if(result.noticias[i].extenso_imagem == ''){
                    noticias += ''
                } else {
                    noticias += ''

                }

                noticias +=
                    '</div>' +
                    '<div class="col-9">' +
                    '<h4>' + result.noticias[i].titulo + '</h4>' +
                    '<p
style="color:gray">' + result.noticias[i].editoria +
('' + result.noticias[i].data_hora.substring(8,10) + '' + result.noticias
[i].data_hora.substring(5,7) + '' -
'' + result.noticias[i].data_hora.substring(11,13) + ':' + result.noticias
[i].data_hora.substring(14,16) + '') + '</p>' +
                    '</div>' +
                    '</div>' +
                    '<br>' +
                    '<hr>'

        }

        // Insere bloco de html no DOM
        $('#noticias').append(noticias)

        // Se nao houver nenhuma noticia cadastrada
    } else {
        $('#noticias').append('<br>Não há notícias
cadastradas<br><br>')
    }
}

/***
 *   Abre uma noticia num modal

```

```
**/
function abrir_noticia(id_noticia) {
    $('#tituloModalNoticia').empty()
    $('#conteudoModalNoticia').empty()

    var noticia = JSON.parse(localStorage.getItem("noticia-
"+id_noticia))
    $('#tituloModalNoticia').html(noticia.titulo)
    $('#conteudoModalNoticia').html(noticia.conteudo)
}
```

Agora que já temos o código do aplicativo, é possível testá-lo via browser, assim como fizemos com o painel administrativo. Basta acessar “localhost/formiga/app/www/” para ver o resultado:

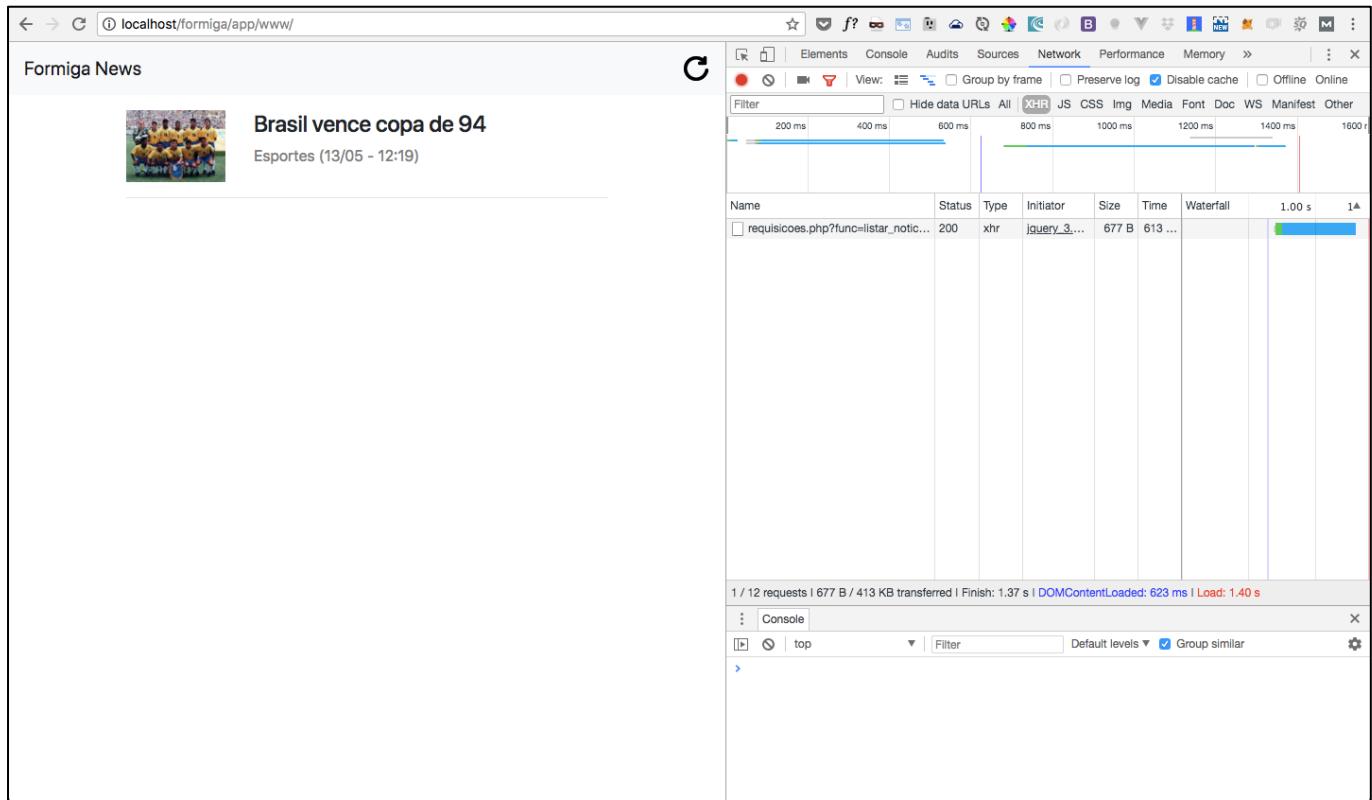


Figura 34 – App sendo testado no navegador

Se desejar ter uma noção mais precisa de como ele ficará num *device* real, o Google Chrome oferece opções de simular diversos aparelhos. No painel do desenvolvedor, clique no ícone que apresenta dois *devices*, um menor e outro maior, no canto superior esquerdo. Em seguida, selecione o *device* que deseja simular, conforme imagem a seguir.

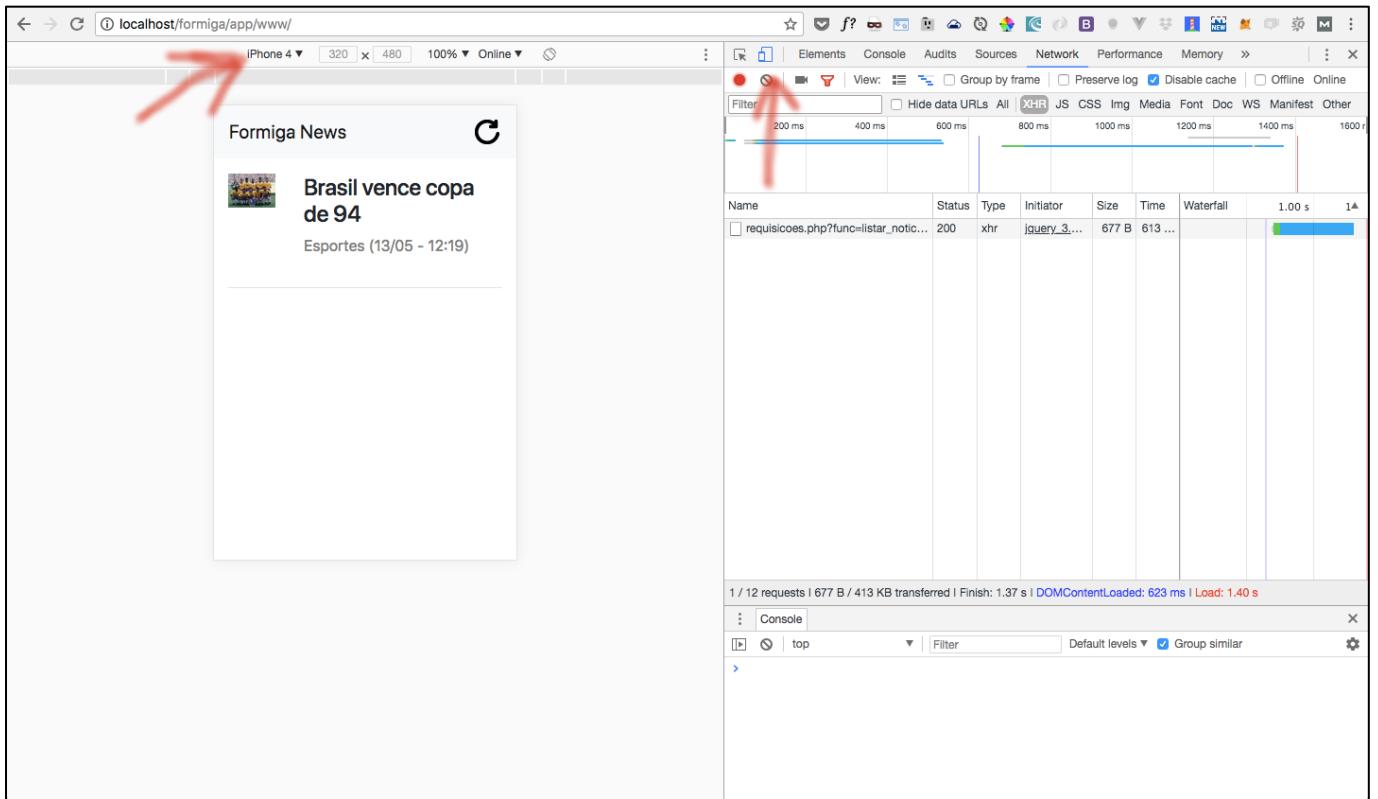


Figura 35 – Navegador simulando um device (iPhone 4)

Essa possibilidade de simular diversos dispositivos é bastante útil para testar a responsividade do aplicativo, se ele se adequa bem aos diversos tamanhos e proporções de tela existentes no mercado. Mesmo sabendo que é praticamente impossível verificar todos os aparelhos disponíveis (há milhares de aparelhos diferentes que rodam Android), recomenda-se testar nos principais. Para o iOS, um bom parâmetro é testar para iPhone 4 ou 5, pois eles possuem a mesma largura de tela, e é a largura que acaba sendo, na maior parte das vezes, a variável “mais frágil”, pois ela tem um papel importante na construção da imagem na tela. No geral, se o aplicativo estiver adequado à largura de tela do iPhone 4/5, ele atenderá bem a maior parte dos dispositivos.

Porém, vale lembrar que não é só de aparência que um *app* é feito. Após garantir que a aparência está ok, é importante testar o *app* em *devices* reais, pois a *engine javascript* que roda no *browser* não é a mesma que roda nos *devices*. As diferenças são pequenas, mas existem. Principalmente, entre a *engine* do Google Chrome e a dos *browsers* iOS. Portanto, recomenda-se compilar o app e testa-lo num *device* real. Uma forma de reduzir a incerteza de defeitos desse tipo é rodar o código também no

navegador Safari, que tem uma *engine javascript* mais parecida com a do *browser iOS mobile*.

Nesse momento, temos tudo funcionando, basta colocar em produção.

### 5.11 Colocando em produção, parte 1: servidor (*backend*) e interface administrativa

Para que nosso serviço esteja disponível para todos os usuários, o código do servidor precisa estar acessível via internet. Além disso, a interface web também precisa estar acessível ao administrador. Portanto, precisamos fazer o upload desse código para o servidor que contratamos anteriormente. Para isso, vamos usar o FileZilla. Basta preencher os dados de conexão (host, usuário e senha) conforme criado anteriormente e fazer a conexão. Uma vez conectado, navegue até a pasta que possui o nome do seu domínio, no nosso caso “formiga.info”, e transfira tanto a pasta “server” quanto a pasta “web” para dentro dela. Não esqueça de atualizar os arquivos de constantes (server/constantes.php e web/constantes.js) antes de fazer o upload, comentando as constantes de desenvolvimento e “descomentando” as constantes de produção.

Um detalhe adicional é que como vamos fazer upload de arquivos para a pasta “uploads”, é preciso alterar a permissão desta pasta para que o PHP possa escrever nela. Com o FileZilla, basta clicar com o botão direito na pasta “uploads” e selecionar as permissões. Em seguida, adicione a permissão de gravar para o grupo, conforme figura a seguir. Também não entraremos em detalhes sobre o funcionamento das permissões de leitura, escrita e execução, mas essa é uma pesquisa válida para o leitor.

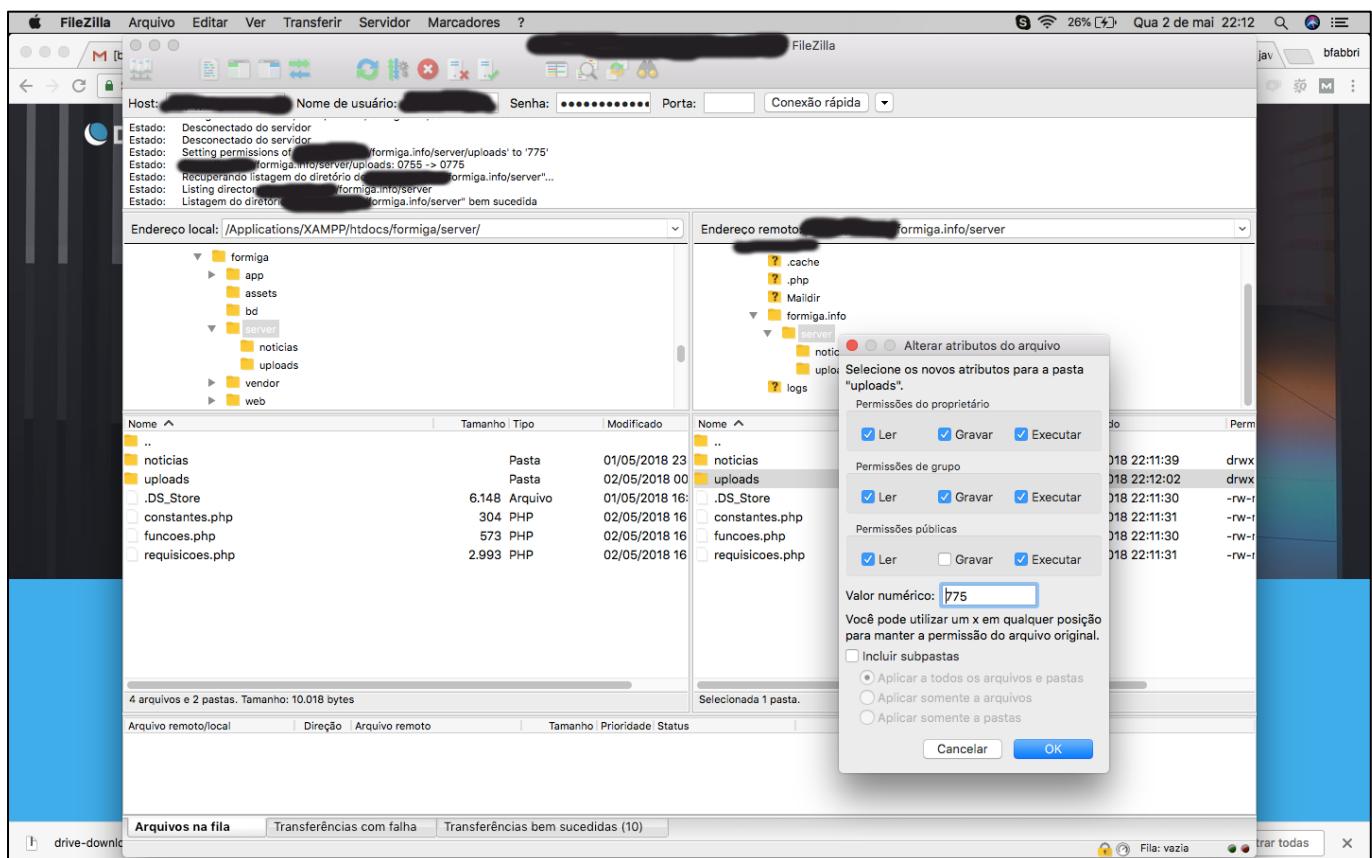


Figura 36 – Alterando as permissões da pasta uploads

Com as pastas “server” e “web” no servidor, já é possível acessar e interagir com o painel administrativo pela internet, no nosso caso, através do endereço [www.formiga.info/web/noticias/noticias.html](http://www.formiga.info/web/noticias/noticias.html). Atente-se para a estrutura de pastas novamente, pois é comum acontecer confusões com isso. Inicialmente, será exibida uma mensagem dizendo que não existem notícias cadastradas, pois agora estamos acessando um banco de dados diferente do que acessamos anteriormente para cadastrar a primeira notícia de teste. Veja na barra de endereços que agora estamos apontando para o domínio que foi comprado, [formiga.info](http://formiga.info), ao invés de [localhost](http://localhost).

The screenshot shows a web-based administrative interface. On the left, there's a form for creating a new news item. It includes fields for 'Editoria' (with a dropdown menu showing 'Editoria'), 'Título' (with a text input field containing 'Título da notícia'), 'Conteúdo' (with a large text area containing 'Conteúdo da notícia'), and 'Imagen' (with a file selection button 'Escolher arquivo' and a message 'Nenhum arquivo selecionado'). On the right, there's a section titled 'Notícias' with the message 'Não há notícias cadastradas'. At the bottom, there's a blue button labeled 'Criar notícia'.

Figura 37 – Painel administrativo em produção

### 5.12 Colocando em produção, parte 2: app

Para compilar o código do aplicativo que geramos e transformá-lo em instaláveis para Android e iOS vamos utilizar o PhoneGap Build (<https://build.phonegap.com/>), um serviço online da Adobe que tira boa parte da complexidade da compilação.

Crie uma conta na versão gratuita do serviço, em seguida gere uma chave para Android, a partir do passo a passo contido da documentação do próprio PhoneGap: <http://docs.phonegap.com/phonegap-build/signing/android/>

Vamos nos conter apenas à geração da chave para Android neste guia por questões de facilidade e custo, pois para assinar aplicativos para a appstore, da Apple, é preciso pagar uma anuidade de 99 dólares para participar do programa de desenvolvedores da Apple.

Agora, com a chave criada e destravada, navegue até a pasta “app”, do XAMPP, selecione o arquivo config.xml e a pasta “www” e comprima as duas de forma a gerar um arquivo .zip. Crie um novo app no PhoneGap Build e suba este arquivo .zip para ser compilado. Em alguns segundos (ou minutos) seu instalável para Android (.apk) estará pronto, conforme figura a seguir:

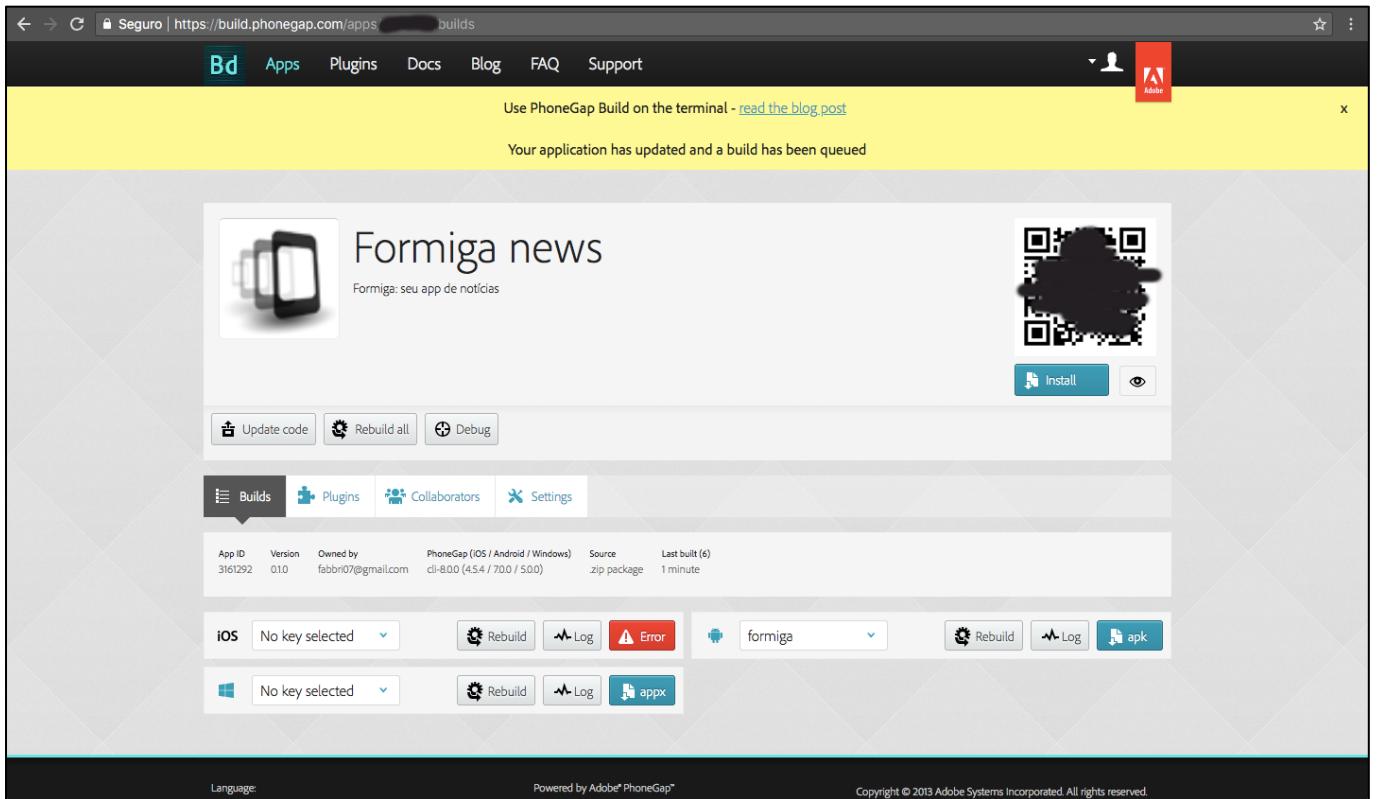


Figura 38 – Aplicativo Android (.apk) criado no PhoneGap Build

Para iOS, o processo seria o mesmo. Ainda, na figura, é possível notar que ocorreu também a compilação para Windows (.appx), mas não vamos entrar neste mérito, pois, muitas vezes, os aplicativos são criados apenas para Android e iOS.

Agora, a alternativa mais simples é utilizar um leitor de QRcode para baixar o *app* diretamente para seu *device* Android. Talvez seja necessário autorizar a instalação em suas configurações, por se tratar de um *app* baixado por fora da Play Store. Caso isso ocorra, o passo a passo é o seguinte:

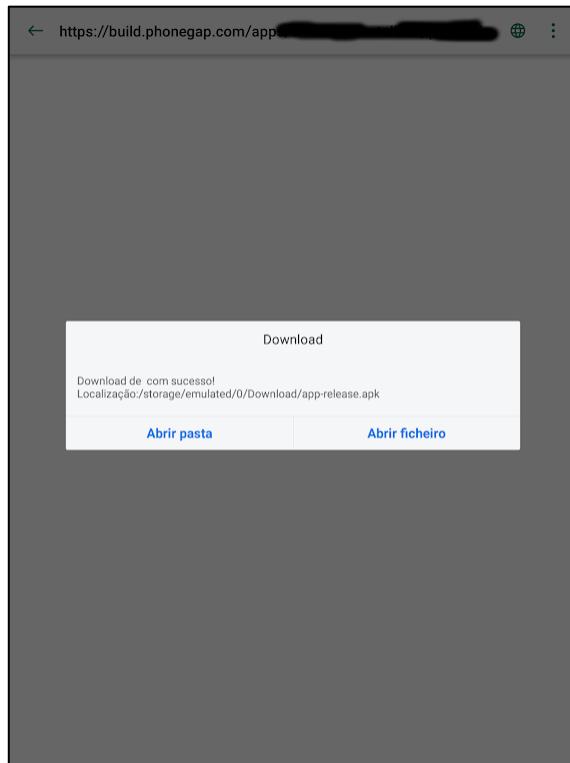


Figura 39 - Instalando app no Android - passo 1

Após o *download*, clique em “Abrir pasta”, conforme mostra a figura 39 acima.

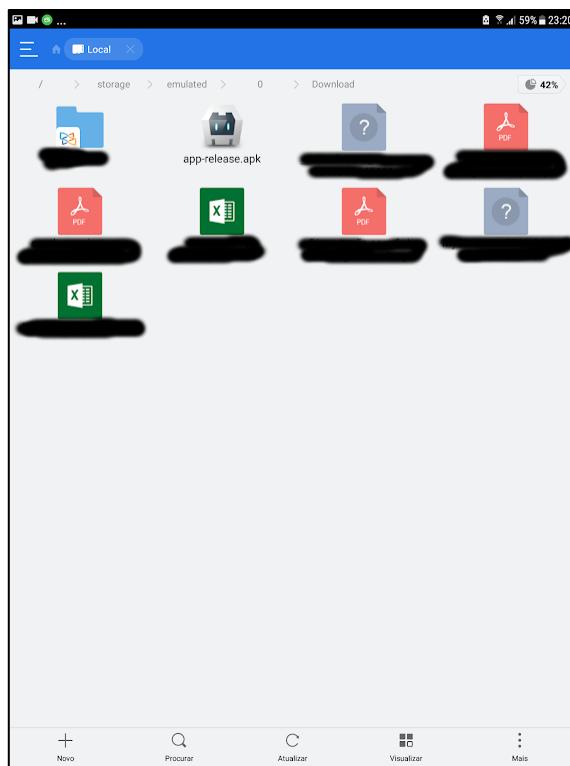


Figura 40 - Instalando app no Android - passo 2

Encontre o *app* na pasta e clique nele. Esse ícone é o ícone padrão que o *phonegap* utiliza quando não existe nenhum ícone definido.

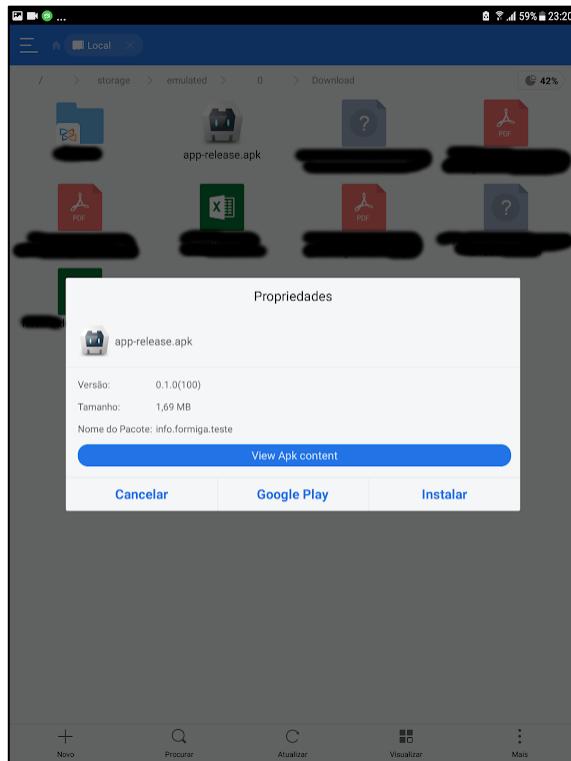


Figura 41 - Instalando app no Android - passo 3

Clique em “Instalar”, conforme figura 41 acima.

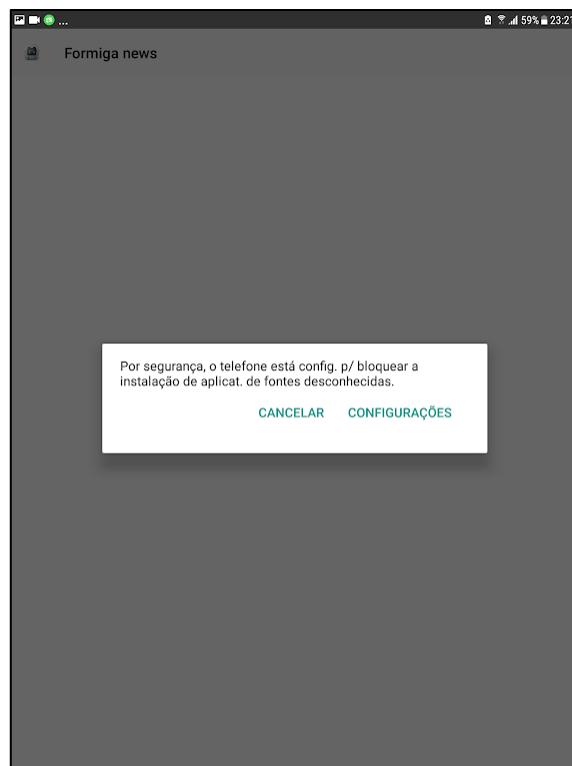


Figura 42 - Instalando app no Android - passo 4

Se a mensagem de segurança aparecer, clique em “CONFIGURAÇÕES”.

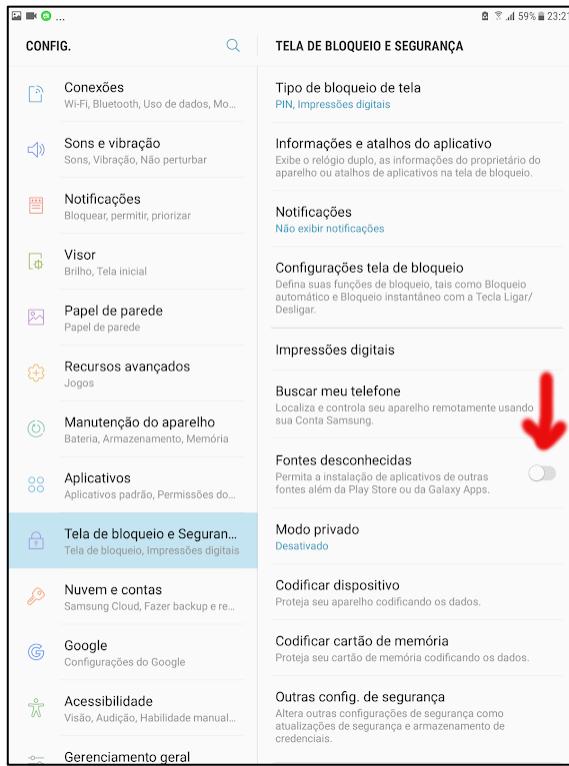


Figura 43 - Instalando app no Android - passo 5

Permita que sejam instalados apps que não são da *Play Store* ou *Google Apps*.

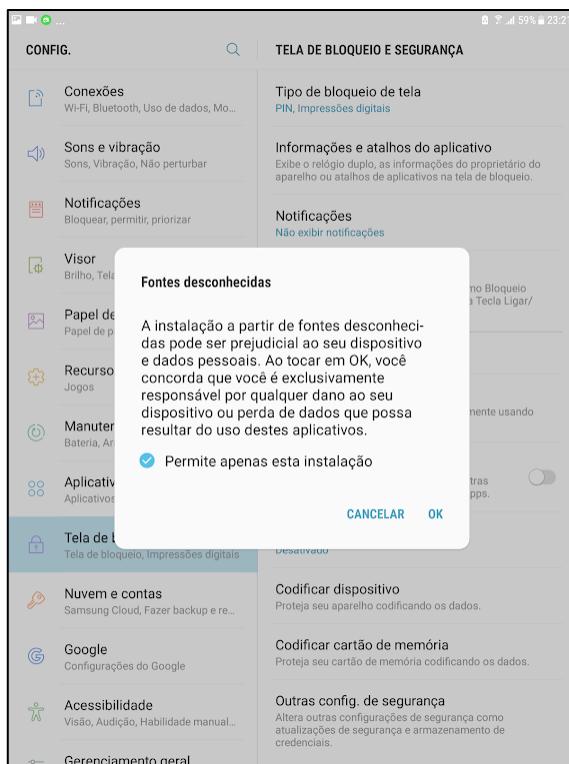


Figura 44 - Instalando app no Android - passo 6

Escolha se essa permissão deve acontecer apenas para essa instalação ou para todas as posteriores e clique em “OK” (recomenda-se permitir apenas esta).

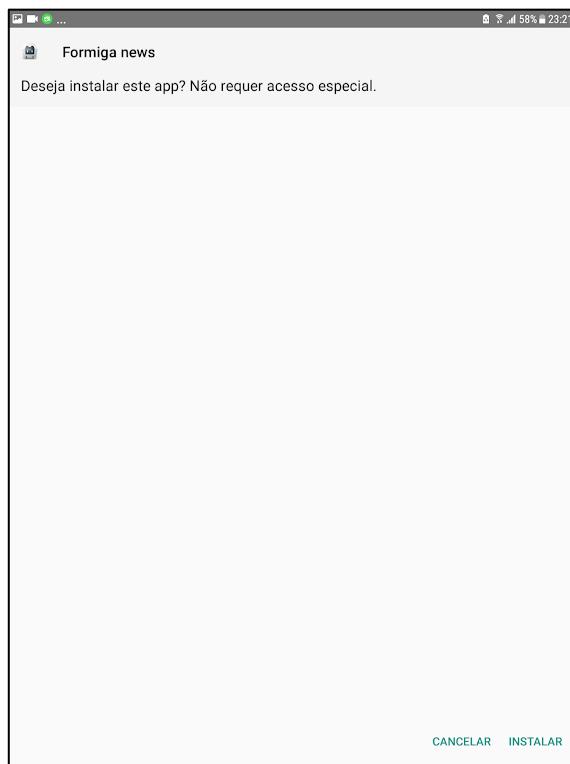


Figura 45 - Instalando app no Android - passo 7

Clique em “INSTALAR”, no canto inferior direito, conforme figura 45 acima.

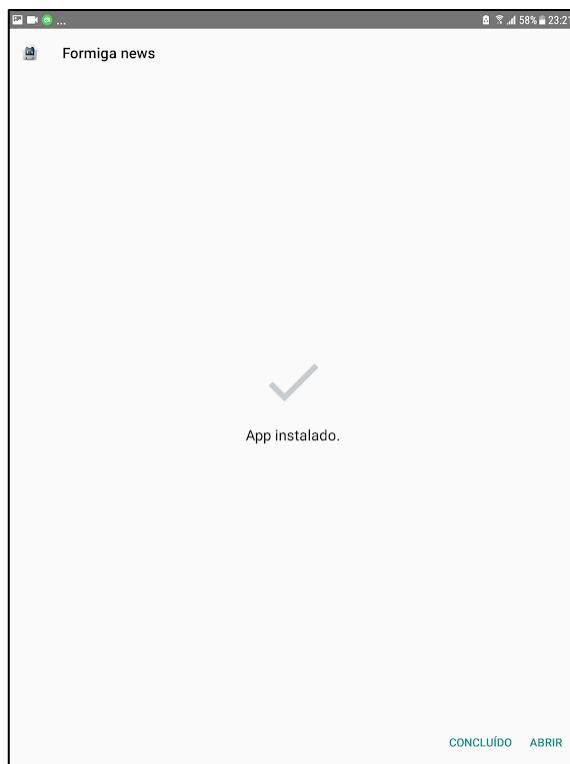


Figura 46 - Instalando app no Android - passo 8

Após a instalação, clique em “ABRIR”, no canto inferior direito.

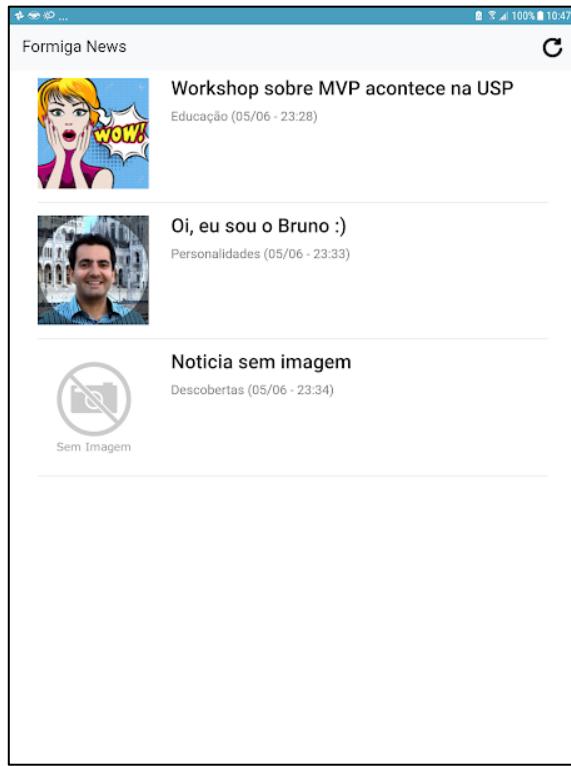


Figura 47 - Instalando app no Android - passo 9

Pronto! Temos um MVP de um serviço de notícias para um jornal local composto de painel administrativo para cadastro de notícias e de um aplicativo para os leitores consultarem as novidades.