****

**INSTITUTO POLITÉCNICO DE LEIRIA**

Gfast

RELATÓRIO DE projeto em sistemas de informação

Relatório de Projeto em Sistemas de Informação para cumprimento dos requisitos necessários à realização da prova de apresentação de projeto do Curso Técnico Superior Profissional (TeSP) de **Programação de Sistemas de Informação** realizado sob a orientação de **Alexandre Frazão do Rosário**

Alexandre Levchenko nº2201155

Duarte Frazão Pereira nº2190715

Programação em Sistemas de informação

18/02/2022

**DECLARAÇÃO**

Declaro que este Relatório se encontra em condições de ser apreciada (o) pelo júri a designar.

O estudante Alexandre Levchenko,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Leiria, 18 de fevereiro de 2022

Relatório de Projeto em Sistemas de Informação para cumprimento dos requisitos necessários à realização da prova de apresentação de projeto do Curso Técnico Superior Profissional (TeSP) de **Programação de Sistemas de Informação** realizado sob a orientação de **Alexandre Frazão do Rosário**

**DECLARAÇÃO**

Declaro que este Relatório se encontra em condições de ser apreciada (o) pelo júri a designar.

O estudante Duarte Frazão Pereira,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Leiria, 18 de fevereiro de 2022

# Agradecimentos

Queremos deixar um agradecimento especial aos professores das UC’s de Acesso Móvel a Sistemas de Informação (Sónia Luz e David Safadinho), Plataformas e Sistemas de Informação (Carlos Ferreira), Serviços e Interoperabilidade de Sistemas (Nuno Simões) e Projeto em Sistemas de Informação (Alexandre Frazão) pelos ensinamentos que nos foram fornecidos ao longo do semestre.

# Resumo

**[RELATÓRIO DE PROJETO - GFAST]**

**[Alexandre Levchenko - 2201155]**

**[Duarte Frazão Pereira - 2190715]**

No âmbito da disciplina de Projeto em Sistemas de Informação do Curso Técnico Superior de Programação de Sistemas de Informação, foi criado o presente relatório com o objetivo de especificar todo o tipo de funcionalidades criadas no projeto desenvolvido.

Ao longo deste projeto foi desenvolvida uma loja virtual de guitarras denominada de “*GFast,* com um *Website* que permite ao utilizador registar-se na aplicação, fazer *login,* e efetuar inúmeras ações. De facto, o utilizar pode ver e comparar centenas de guitarras,marcas e categorias, editar o seu perfil, adicionar e remover guitarras aos favoritos e ao carrinho de compras, e por fim, adicionar, remover e alterar avaliações às guitarras.

Este projeto possui também uma aplicação móvel, onde é possível o utilizador registar-se, fazer *login,* ver e comparar centenas de guitarras, editar perfil, adicionar e remover guitarras favoritas.

Antes de iniciar todo o processo de implementação de código, foi realizado um planeamento com todas as tarefas, tal como a dificuldade e o tempo previsto para a realização de cada uma. Contudo, apesar de maior parte das tarefas terem sido realizadas, nem sempre o que foi previsto coincidiu com a realidade.

Um dos objetivos com este projeto era a aquisição de conhecimentos técnicos por parte dos membros do grupo, através da criação de um website e da respetiva aplicação móvel, que proporcione aos clientes uma experiência agradável na compra de guitarras *online*, oferecendo-lhe preços imbatíveis e simplicidade durante a compra.

.

PALAVRAS-CHAVE: guitarras, gfast, projeto, curso, compra

# Índices

**Índice Principal**

[Agradecimentos iv](#_Toc96114653)

[Resumo v](#_Toc96114654)

[Índices vii](#_Toc96114655)

[1. Introdução 2](#_Toc96114656)

[2. Metodologia 4](#_Toc96114657)

[3. Arquitetura do Sistema 5](#_Toc96114658)

[4. Gestão do Projeto 6](#_Toc96114659)

[4.1. Base de dados 6](#_Toc96114660)

[4.2. Aplicação Móvel 7](#_Toc96114661)

[4.3. Website 8](#_Toc96114662)

[4.4. API 9](#_Toc96114663)

[5. Análise 10](#_Toc96114664)

[5.1. Tema do Projeto 10](#_Toc96114665)

[5.2. Objetivos Principais 10](#_Toc96114666)

[5.3. Requisitos e funcionalidades 10](#_Toc96114667)

[5.3.1. Website (frontend) 10](#_Toc96114668)

[5.3.2. Website (backend) 11](#_Toc96114669)

[5.3.3. Aplicação Móvel 11](#_Toc96114670)

[5.3.4. API 11](#_Toc96114671)

[5.4. Análise Concorrencial 12](#_Toc96114672)

[5.4.1. LudiMusic 12](#_Toc96114673)

[5.4.2. Thomann 13](#_Toc96114674)

[5.4.3. ESP 14](#_Toc96114675)

[6. Desenho 16](#_Toc96114676)

[6.1. *Mockups* 16](#_Toc96114677)

[6.2. Modelo de dados 16](#_Toc96114678)

[7. Implementação 17](#_Toc96114679)

[7.1. Conceitos e Tecnologias utilizados 17](#_Toc96114680)

[7.2. Base de dados 17](#_Toc96114681)

[7.3. Aplicação *Web* 18](#_Toc96114682)

[7.3.1. MVC 18](#_Toc96114683)

[7.3.2. GII 18](#_Toc96114684)

[7.3.3. ACF e RBAC 19](#_Toc96114685)

[7.3.4. Templates 22](#_Toc96114686)

[7.3.5. EXTRAS 23](#_Toc96114687)

[7.4.1. Singleton 25](#_Toc96114688)

[7.4.2. Atividades e Fragmentos 27](#_Toc96114689)

[7.5.1. cURL resultado no CMD 29](#_Toc96114690)

[7.5.2. EndPoints 31](#_Toc96114691)

[1. **Users** 32](#_Toc96114692)

[2. **Guitarras** 32](#_Toc96114693)

[3. **Subcategorias** 32](#_Toc96114694)

[**3.1.1.** **Categorias** 32](#_Toc96114695)

[8. Testes 34](#_Toc96114696)

[8.1. BACKEND 34](#_Toc96114697)

[8.2. FRONTEND 36](#_Toc96114698)

[9. Conclusão e trabalho futuro 40](#_Toc96114699)

[10. Bibliografia 41](#_Toc96114700)

[Anexos 1](#_Toc96114701)

[Anexo A 2](#_Toc96114702)

[Anexo B 2](#_Toc96114703)

[Anexo C 4](#_Toc96114704)

[Anexo D 5](#_Toc96114705)

[Anexo E 6](#_Toc96114706)

[Anexo F 7](#_Toc96114707)

[Anexo G 8](#_Toc96114708)

**Índice de Tabelas**

[Tabela 1 - Requisitos e Funcionalidades (Frontend) 10](#_Toc96098322)

[Tabela 2 - Requisitos e Funcionalidades WEB(Frontend) 11](#_Toc96098323)

[Tabela 3 - Requisitos e Funcionais Aplicação Móvel 11](#_Toc96098324)

[Tabela 4 - Requisitos e Funcionalidades API 11](#_Toc96098325)

[Tabela 5 - Descrição da LudiMusic 12](#_Toc96098326)

[Tabela 6 - Descrição da Thomann 13](#_Toc96098327)

[Tabela 7 - Descrição da ESP 14](#_Toc96098328)

[Tabela 8 - Destaque de algumas tecnologias e conceitos utilizados 17](#_Toc96098329)

[Tabela 9 - Packages da aplicação móvel 24](#_Toc96098330)

[Tabela 10 – Backend: Testes unitários 34](#_Toc96098331)

[Tabela 11 – Backend: Testes Funcionais 35](#_Toc96098332)

[Tabela 12 – Frontend: Testes unitários 36](#_Toc96098333)

[Tabela 13 – Frontend: Testes Funcionais 37](#_Toc96098334)

[Tabela 14 – Frontend: Testes de Aceitação 38](#_Toc96098335)

**Índice de Figuras**

[Figura 1 – Arquitetura do Sistema 5](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114709)

[Figura 2 – Planeamento de Tarefas: Base de Dados 6](#_Toc96114710)

[Figura 3 – Cronologia: Base de Dados 6](#_Toc96114711)

[Figura 4 – Planeamento de Tarefas: Aplicação Móvel 7](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114712)

[Figura 5 – Cronologia: Aplicação Móvel 7](#_Toc96114713)

[Figura 6 – Planeamento de Tarefas: PSI (Web) 8](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114714)

[Figura 7 – Cronologia: PSI (Web) 8](#_Toc96114715)

[Figura 8 – Planeamento de Tarefas: SIS (API) 9](#_Toc96114716)

[Figura 9 – Cronologia: SIS (API) 9](#_Toc96114717)

[Figura 10 – Website da LudiMusic 12](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114718)

[Figura 11 – Website da Thomann 13](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114719)

[Figura 12 – Website da ESP 14](#_Toc96114720)

[Figura 13- Menu Inicial de GII 19](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114721)

[Figura 14 - Esboço das permissões definida para cada Role 20](#_Toc96114722)

[Figura 15 - Exemplo da criação de uma permissão chamada “crudUsers” 20](#_Toc96114723)

[Figura 16 - Exemplo da criação do Role admin 20](#_Toc96114724)

[Figura 17- Exemplo de implementação do RBAC 21](#_Toc96114725)

[Figura 18 - Exemplo de ACF 21](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114726)

[Figura 19 - Template Frontend 22](#_Toc96114727)

[*Figura 20 - Template Backend* 22](#_Toc96114728)

[Figura 21 - Criação de pontos no mapa através de Coordenadas 23](#_Toc96114729)

[Figura 22 - Mapa da localização de lojas 23](#_Toc96114730)

[Figura 23 - Exemplo de Pedido Get utilizando o Volley 26](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114731)

[Figura 24 Exemplo de gestão da bd local 26](#_Toc96114732)

[Figura 25 – implementação da API 28](#_Toc96114733)

[Figura 26 – Pasta de controladores 29](#_Toc96114734)

[*Figura 27 – Exemplo GET Categorias* 30](#_Toc96114735)

[Figura 28 – Exemplo POST Categorias 30](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114736)

[Figura 29 – Exemplo PUT Categorias 31](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114737)

[Figura 30 – Exemplo DELETE Categorias 31](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114738)

[Figura 31 – Código de messaging 33](https://myipleiria-my.sharepoint.com/personal/2190715_my_ipleiria_pt/Documents/Modelo_Relatório_Final%20_ProjetoSistemasInformação%20(2)%20(1).docx#_Toc96114739)

[Figura 32 – testes unitários backend 35](#_Toc96114740)

[Figura 33 – testes funcionais backend 36](#_Toc96114741)

[Figura 34 – Testes unitários frontend 37](#_Toc96114742)

[Figura 35 – Testes funcionais backend 38](#_Toc96114743)

[Figura 36 – Teste complete de aceitação 39](#_Toc96114744)

**Lista de Siglas e Acrónimos**

|  |  |
| --- | --- |
| **API** | Application Programming Interface |
| **CRUD** | Create, Read, Update Delete |
| **CSS** | Cascading Style Sheets |
| **CTESP** | Curso Técnico Superior Profissional |
| **ESTG** | Escola Superior de Tecnologia e Gestão |
| **HTML** | Hyper Text Markeup Language |
| **IPL** | Instituto Politécnico de Leiria |
| **JSON** | JavaScript Object Notation |
| **MVC** | Model-View-Controller |
| **PHP** | Hypertext Preprocessor |
| **SMTP** | Simple Mail Transfer Protocol |
| **SQL** | Structured Query Language |
| **URL** | Uniform Resource Locator |

# Introdução

No presente relatório irá ser especificado todo o processo efetuado ao longo do desenvolvimento dos requisitos necessários, desde o planeamento até à sua conclusão.

O projeto *GFast* é baseado numa loja *online*, na qual é possível realizar compras e pesquisas relativamente às guitarras mais revelantes no momento. Para tal, foi desenvolvido um *Website* e uma aplicação móvel que se encontram interligadas a uma *API*, de forma a existir comunicação entre ambas.

Inicialmente, foi efetuado um planeamento, através do qual foram estipuladas as diferentes tarefas para cada membro do grupo, assim como a sua respetiva dificuldade, duração e data para a sua realização. No planeamento foi utilizada uma metodologia “Scrum”, com “Sprints” de duração de 2 semanas, para garantir o tipo e a qualidade do produto final.

É importante mencionar que devido à desistência de um membro do grupo, todos os objetivos e divisão de tarefas foram reformulados, para que fosse possível concluir todos os requisitos dentro dos prazos estipulados.

Após o planeamento, foi criada uma base de dados com tabelas e ligações, através de chaves estrangeiras que armazena todas as informações necessárias para que o projeto tenha sempre a maior segurança e o melhor desempenho.

Ao longo do desenvolvimento, surgiram constantes alterações nas tabelas e nas ligações, de forma a adicionar funcionalidades e para completar mais o programa previamente planeado.

Contudo, cada sistema tem algumas finalidades únicas que foram implementadas.

No *frontend* do *Website*, cada utilizador tem a oportunidade de adicionar uma avaliação sobre cada guitarra inserida, enquanto no *backend* somente um administrador ou gestor pode adicionar as guitarras, marcas, categorias ou subcategorias.

Ao efetuar o registo ou *login* na aplicação *web*, o utilizador tem a possibilidade de editar o seu próprio perfil, desativar a sua conta, adicionar e remover as guitarras aos favoritos e ao carrinho de compras. É-lhe possível ainda inserir, editar e eliminar as suas próprias avaliações da guitarra, e, por fim, também pode realizar uma encomenda.

Quanto à aplicação móvel, é possível ao utilizador realizar o seu registo, fazer *login,* editar o seu próprio perfil, e por fim adicionar e remover guitarras aos favoritos.

*GFast* dispôs também como foco principal a implementação de um espaço digital, com bastante simplicidade e de fácil navegação para o usuário.

Em ambos os sistemas, o cliente pode realizar o seu registo com *username*, email, nome, apelido, contacto, cidade, contribuinte e *password*, sendo que quando efetua o registo, basta o *username* e *password* para efetuar o *login*.

Em termos de requisitos funcionais, tem-se que o usuário poderá editar o seu perfil, no caso de existência de algum erro relativamente à informação presente na sua conta. Adicionalmente, há a possibilidade de carregar numa guitarra, que se encontre na lista de guitarras, e visualizar os seus detalhes e as especificidades.

Perante uma grande dimensão de guitarras, o cliente tem a oportunidade de adicionar as que lhe despertaram interesse nos favoritos, no sentido de a guardar para não as “perder de vista”, oferecendo-lhe uma mais-valia ao navegar na aplicação e no *website*. Em sentido contrário, poderá, posteriormente, removê-la dos favoritos.

A API desenvolvida neste projeto serve de ponte entre a aplicação móvel e o *website*, enviando dados a partir do *website* e recebendo os dados dos utilizadores provenientes da aplicação móvel.

Neste presente relatório iremos abordar vários aspetos, tais como:

* Metodologia adotada no projeto;
* Arquitetura do sistema, com todas as principais componentes tecnológicas e a sua interação;
* Gestão do projeto, principais objetivos, temática, e a análise dos requisitos e das funcionalidades dos sistemas;
* Todos os desenhos, especificando os protótipos, casos de uso, modelo de dados e *mockup’s;*
* Implementação geral;
* Todos os testes realizados;
* Conclusão, com a nossa opinião e visão para o futuro.

# Metodologia

De forma a organizar melhor o desenvolvimento do projeto foi utilizada uma metodologia Scrum, uma metodologia considerada ágil e capaz de prever antecipadamente, diminuindo simultaneamente o risco.

Inicialmente, foram definidas as tarefas previstas no *backlog*, uma vez que é aqui onde se encontram todas as tarefas a serem efetuadas ordenadamente, com vista a obter o projeto final.

A partir das tarefas previamente definidas no backlog forem feitas sprints de duas em duas semanas, nas quais eram definidas as tarefas a completer na próxima sprint, podendo estas ser um User Story, uma feature ou um problema.

Graças ao facto de um dos elementos do grupo ter desistido, levou a que tivessemos de recorrer a métodos de realização das tarefas diferentes e menos organizados. Contudo, foram realizadas as reuniões quinzenais para se definir o que seria realizado na próxima sprint.

No que concerne à Scrum team, tem-se que o Scrum Master foi considerado o professor Alexandre Rosário, e como Cliente os Júris do projeto final de curso do TeSP em PSI. Na equipa de desenvolvimento encontram-se os estudantes Alexandre Levchenko e Duarte Pereira, sendo que o este ultimo é também o Product Owner.

# 

# Arquitetura do Sistema

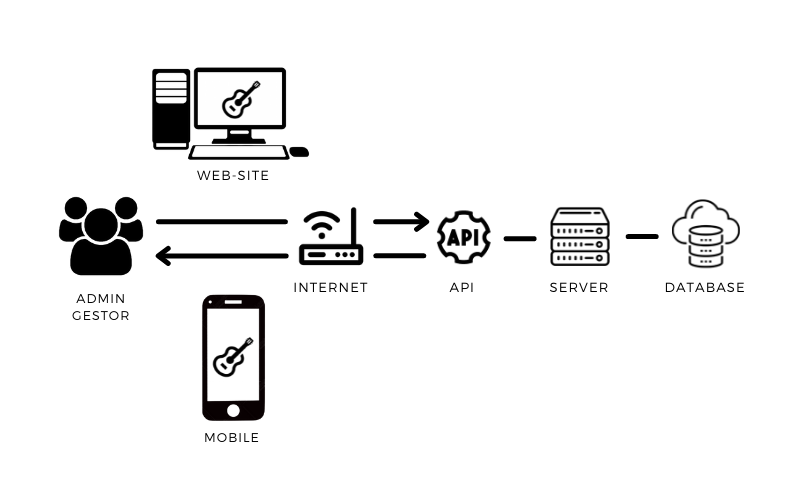
A arquitetura no nosso sistema é constituída por uma API *Rest* que é utilizada como ponte na junção da Aplicação Android e o nosso Website, sendo que o Website é dividido em *Frontend,* para os utilizadores normais, e em *Backend*, para utilizadores apropriados para acederem a essa área.

A parte do *Frontend* disponibiliza aos nossos utilizadores uma interface bastante apelativa e com um design bastante simples para facilitar na navegação, enquanto o *Backend* possibilita a parte de a administração realizar a gestão de conteúdo inserido no *Frontend*.

Temos também a nossa Aplicação Móvel que realiza a comunicação entre HTTP e a API *Rest*, para obter todas as informações necessárias, assim como, por exemplo, o *login*, o registo e as guitarras inseridas. O nosso Website efetua a comunicação com a API localmente, devido a estarem no mesmo servidor.

Neste projeto foi desenvolvida uma base de dados através do MySQL para gerir todos os dados colocados e retirados.

Para efetuar a demonstração do nosso sistema, realizamos dois desenhos para ser bem visível todos os pormenores. ([Anexo B](#_Anexo_B) e figura 1)



Fonte: Elaboração Própria

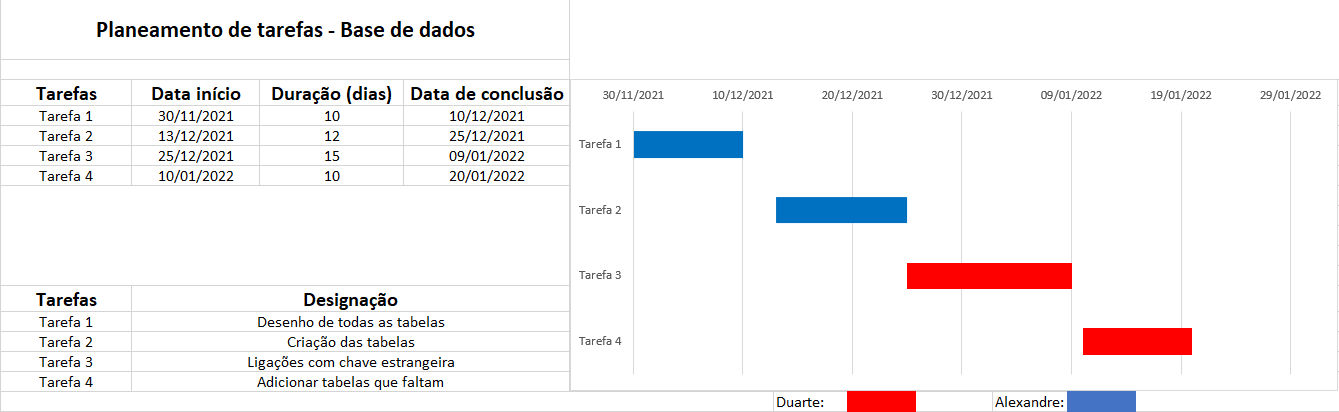
Figura 1 – Arquitetura do Sistema

# Gestão do Projeto

De modo a gerir melhor as tarefas e tempo ao longo deste projeto, foi atribuído a cada membro um conjunto de tarefas, assim como um período de tempo específico. Uma vez efetuadas as tarefas, foram demonstradas entre os membros a forma de realização das mesmas, com o intuito de cada um ter uma noção do que foi desenvolvido.

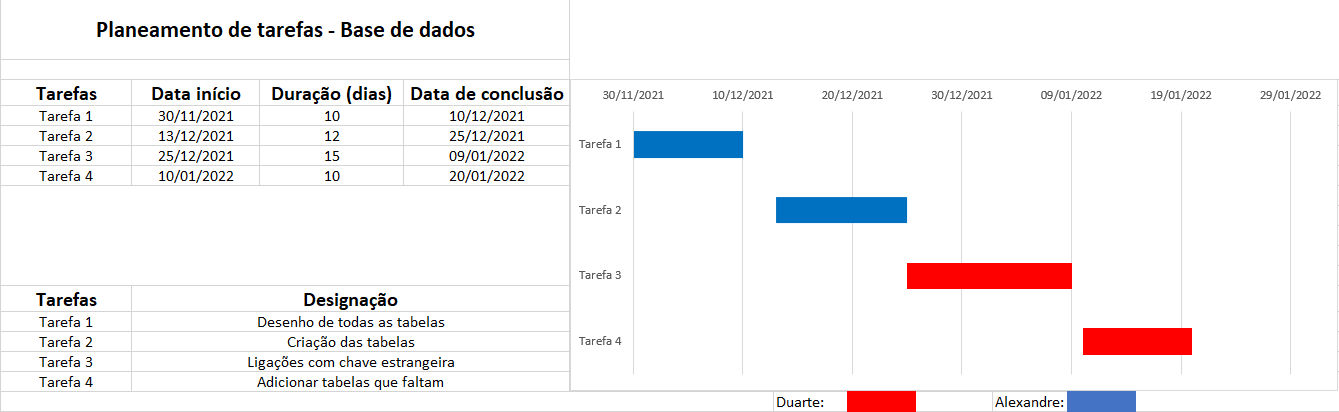
## Base de dados

Figura 2 – Planeamento de Tarefas: Base de Dados



Fonte: Elaboração Própria

Figura 3 – Cronologia: Base de Dados



Fonte: Elaboração Própria

## Aplicação Móvel

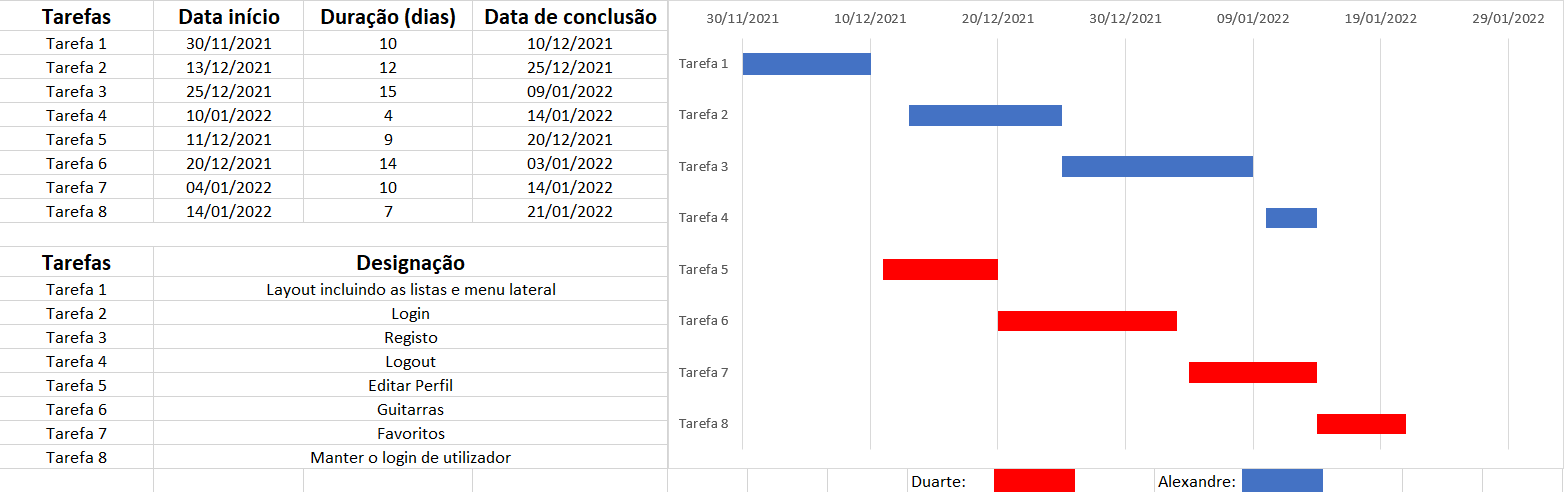
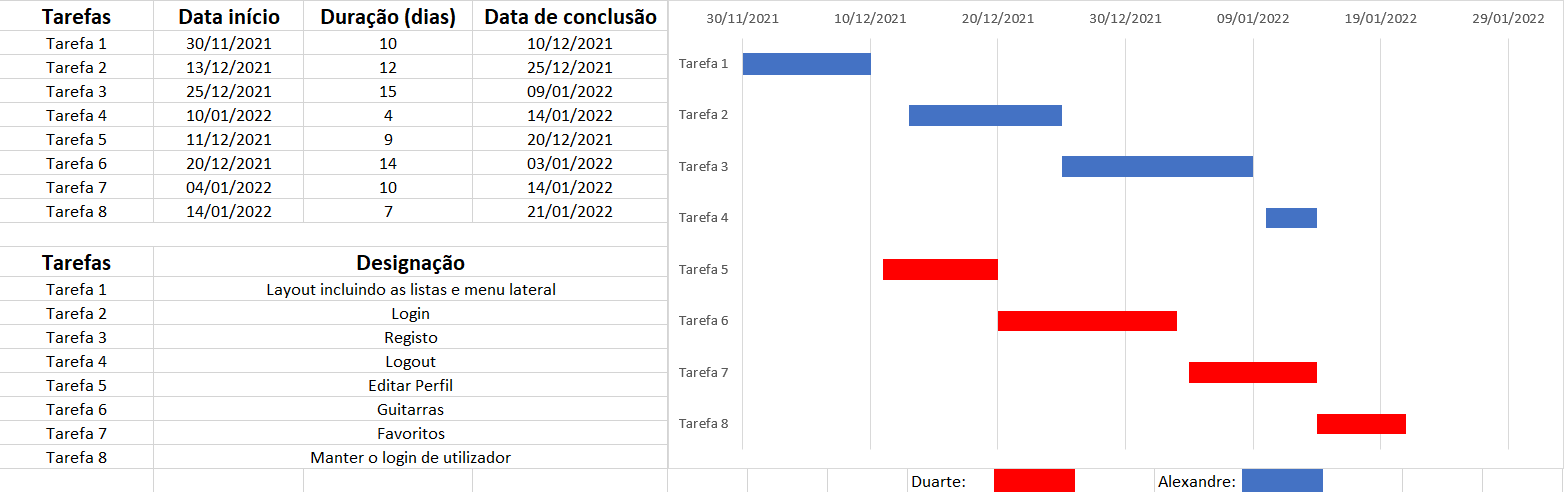


Figura 4 – Planeamento de Tarefas: Aplicação Móvel

Fonte: Elaboração Própria

Figura 5 – Cronologia: Aplicação Móvel



Fonte: Elaboração Própria

## Website

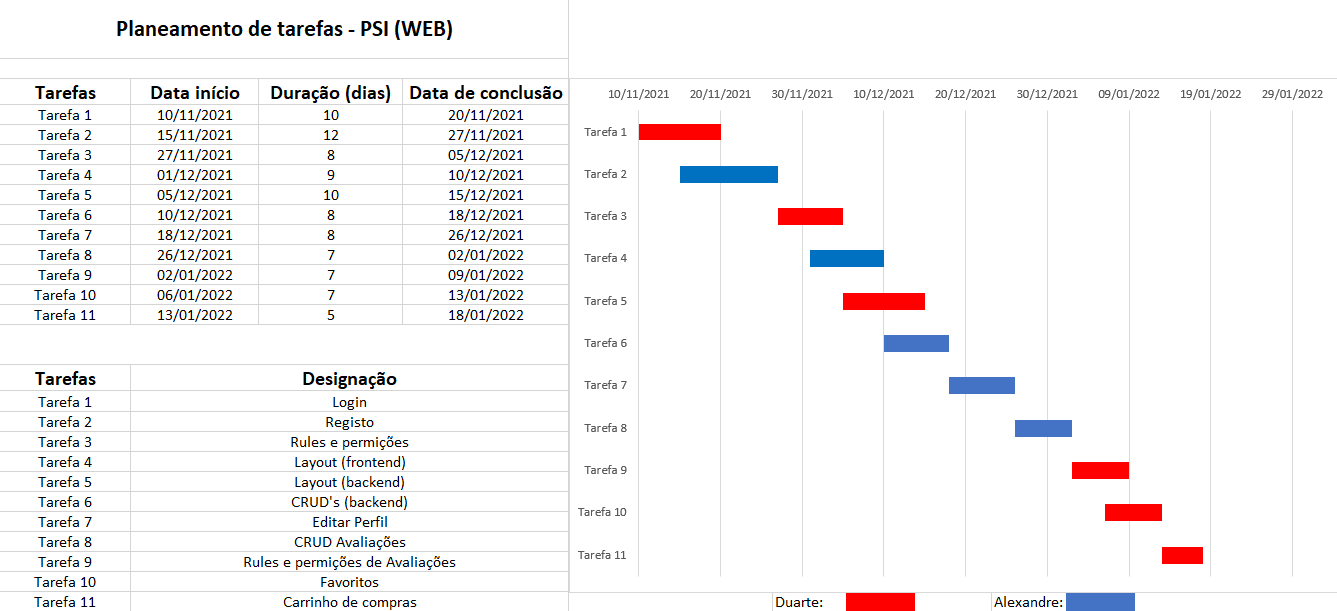
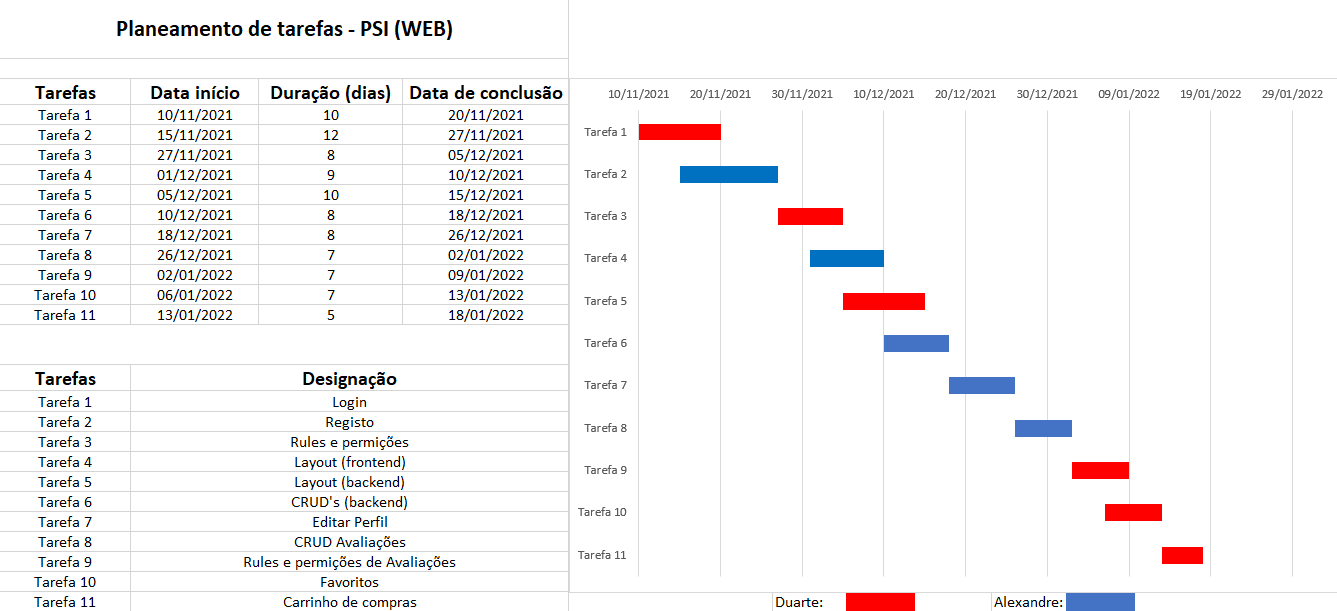


Figura 6 – Planeamento de Tarefas: PSI (Web)

Fonte: Elaboração Própria

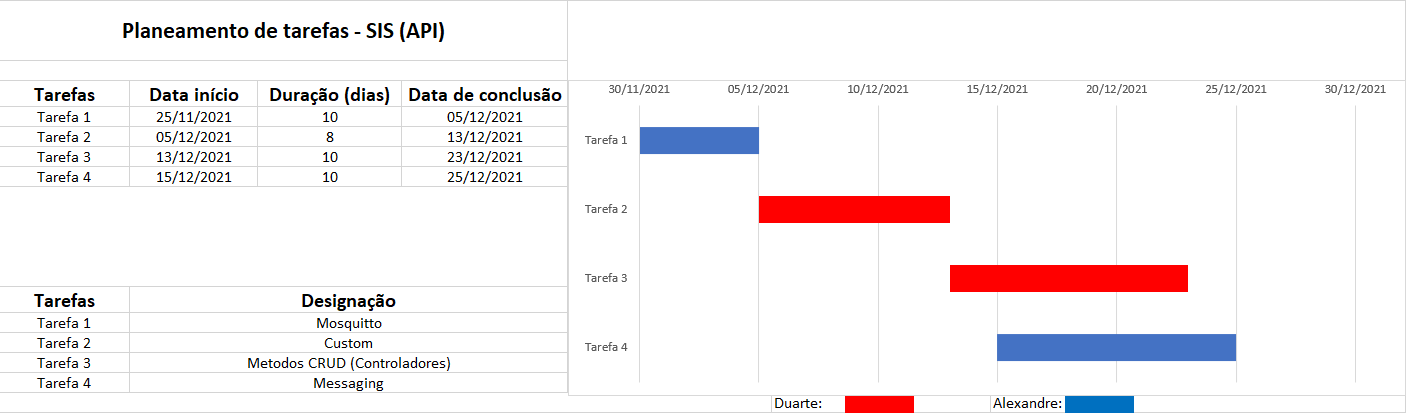
Figura 7 – Cronologia: PSI (Web)



Fonte: Elaboração Própria

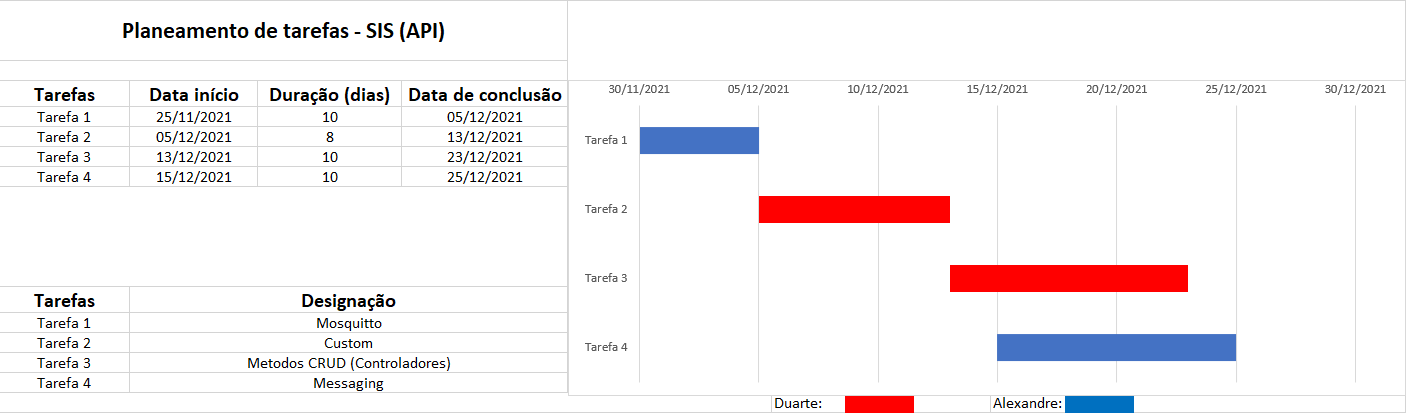
## API

Figura 8 – Planeamento de Tarefas: SIS (API)



Fonte: Elaboração Própria

Figura 9 – Cronologia: SIS (API)



Fonte: Elaboração Própria

# Análise

## Tema do Projeto

Desde o seu início, o projeto foi criado com o intuito de melhorar o processo de venda em todas as lojas no mundo virtual, pelo que não só irá servir de base de comparação no caso de lojas de guitarras, mas também com lojas que atuam noutros mercados, tais como, por exemplo, telemóveis, carros, entre outros.

Com o decorrer deste projeto, foram-se sempre somando os objetivos para que esta loja conseguisse ser um exemplo para o mercado mundial.

Assim, fomos melhorando as nossas ideias, e cada vez mais aumentámos o nosso foco para que este sistema fosse algo que teríamos todo o gosto de utilizar no dia a dia.

## Objetivos Principais

O nosso objetivo principal residiu no desenvolvimento de uma aplicação e *website* que disponha de todas as funcionalidades que uma loja deverá ter, como, por exemplo, o *login*, o registo e informação detalhada de cada produto inserido.

Pretendemos também desenvolver algo que fosse facilmente navegável, com menus simples e, adicionalmente, um design agradável, promovendo, deste modo, a imagem da “GFast”.

De modo a satisfazer mais a pesquisa dos clientes, há a oportunidade de efetuar pesquisas, adicionar/remover artigos nos favoritos, assim como recorrer à ferramenta de edição do perfil, na eventualidade de existência de erros acerca das informações do cliente.

## Requisitos e funcionalidades

## Website (frontend)

Tabela 1 - Requisitos e Funcionalidades (Frontend)

|  |  |
| --- | --- |
| RF-01 | Ver e adicionar avaliações nos produtos |
| RF02 | Realizar *logout*, login e registo |
| RF-03 | Editar o próprio perfil |
| RF-04 | Adicionar ao carrinho de compras e efetuar a encomenda |
| RF-05 | Realizar pesquisas por categorias, subcategorias, marcas e produtos |
| RF-06 | Adicionar ou remover uma guitarra aos favoritos |
| RF-07 | Ver os detalhes e especificações de cada guitarra |

## Website (backend)

Tabela 2 - Requisitos e Funcionalidades WEB(Frontend)

|  |  |
| --- | --- |
| RF-BO-01 | Realizar *logout*, login e registo |
| RF-BO-02 | Gerir perfis de funcionários/gestores de cada loja |
| RF-BO-03 | Gerir as marcas, categorias e subcategorias dos produtos existentes nas lojas |
| RF-BO-04 | Gerir guitarras |

## Aplicação Móvel

Tabela 3 - Requisitos e Funcionais Aplicação Móvel

|  |  |
| --- | --- |
| RF-01 | Ver os detalhes e especificações de cada guitarra |
| RF02 | Realizar *logout*, login e registo |
| RF-03 | Editar o próprio perfil |
| RF-04 | Realizar pesquisas de guitarras |
| RF-05 | Adicionar ou remover uma guitarra aos favoritos |

## API

Tabela 4 - Requisitos e Funcionalidades API

|  |  |
| --- | --- |
| RF-01 | Atualização de guitarras |
| RF02 | Gestão da autenticação |

## Análise Concorrencial

## LudiMusic

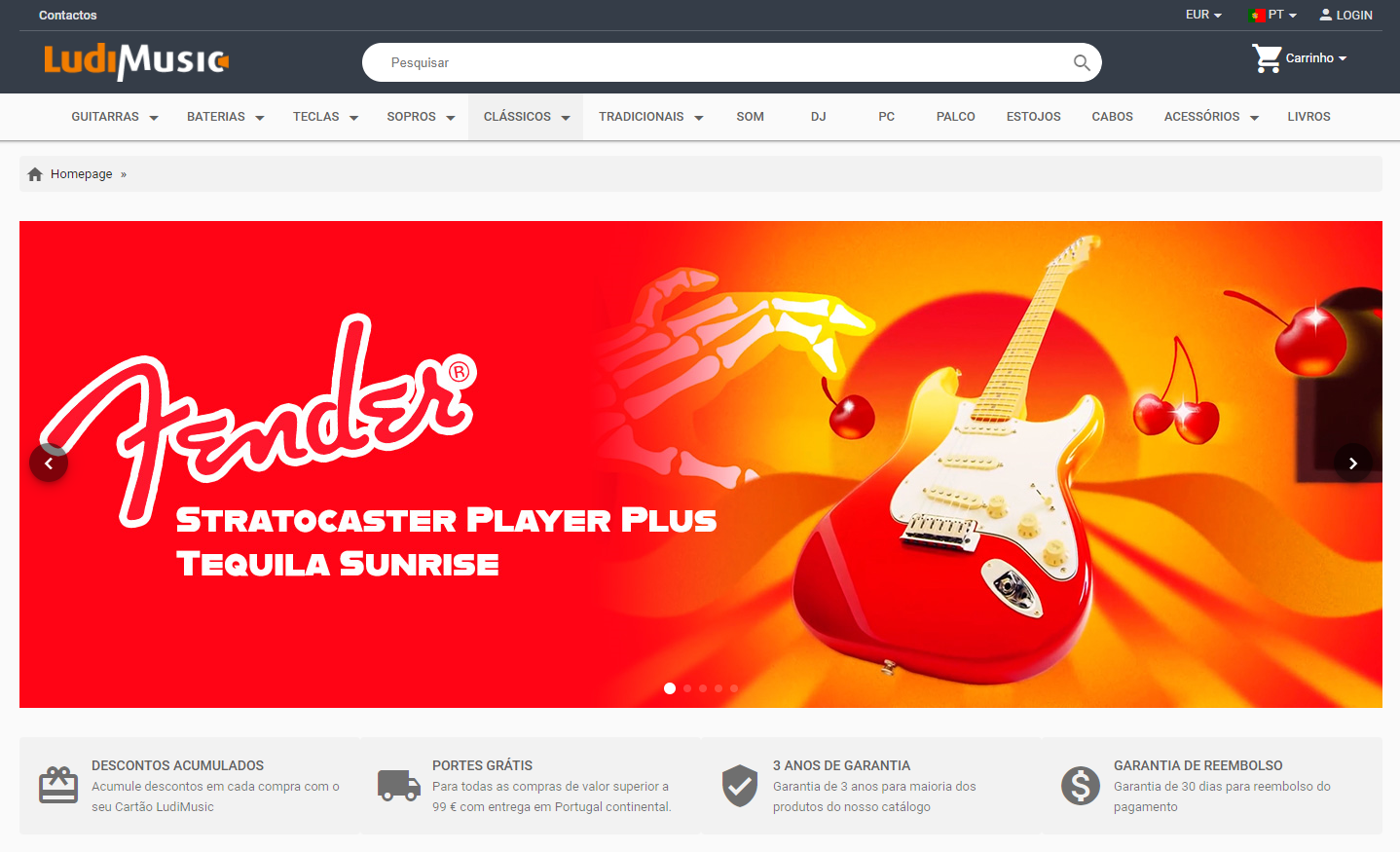


Figura 10 – Website da LudiMusic

Fonte: Website da LudiMusic

Tabela 5 - Descrição da LudiMusic

|  |  |
| --- | --- |
| Nome: | LudiMusic (Ludimusic, 2022) |
| Site: | https://www.ludimusic.com/ |
| Descrição: | A LudiMusic é uma loja de música, com sede em Leiria, e com uma filial no porto. Atualmente possui uma vasta loja online com diversas categorias de instrumentos e acessórios musicais. |
| Vantagens: | - Variedade para além das Guitarras  - *Website* responsivo  - Interface do website intuitiva  - Sistema de pontos |
| Desvantagens: | - Demasiadas categorias  - Falta de conteúdo exclusivo  - Sem aplicação móvel |
| O que falta: | - Aplicação móvel  - Conteúdos exclusivos  - Mapa com lojas física |

Fonte: Elaboração Própria

## Thomann

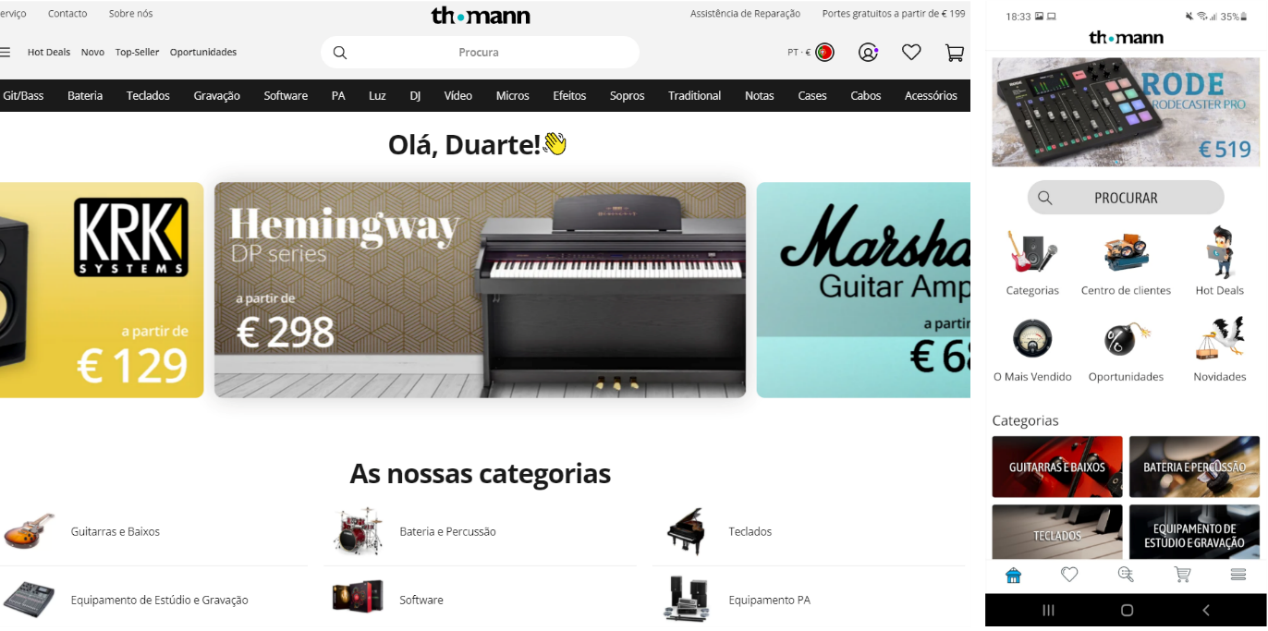


Figura 11 – Website da Thomann

Fonte: Website da Thomann

Tabela 6 - Descrição da Thomann

|  |  |
| --- | --- |
| Nome: | Thomann (Thomann, 2022) |
| Site: | Web <https://www.thomann.de/pt/index.html> Android https://play.google.com/store/apps/details?id=de.thomann |
| Descrição: | A Thomann é uma loja alemã que vende todo o tipo de material do mundo da música, e possui um website que serve de loja online, assim como uma aplicação móvel que facilita a experiência da compra online para o utilizador. |
| Vantagens: | - Aplicação móvel responsiva  - Preços baixos  - Sem conteúdo exclusivo  - Favoritos  - “*Hot Deals*” |
| Desvantagens: | - Falta de um sistema de pontos  - Demasiadas categorias de produtos  - Falta de sistema QR para interagir com o *Website* |
| O que falta: | - Sistema de pontos, que motiva o cliente a comprar na loja  - Mapa com lojas físicas  - Código QR, para que o website interaja com o Android.  - Localização de lojas |

Fonte: Elaboração Própria

## ESP

Figura 12 – Website da ESP



Fonte: Website da ESP

Tabela 7 - Descrição da ESP

|  |  |
| --- | --- |
| Nome: | ESP Guitar Company (ESP, n.d.) |
| Site: | https://www.espguitars.com/ |
| Descrição: | A ESP é uma marca de guitarras americana que possui uma loja online para venda de seus produtos. |
| Vantagens: | - *Website* responsivo  - Interface do *website* intuitiva  - Apenas vendem guitarras, o que simplifica o processo de compra no *website*  - Galeria de fotos  - Notícias |
| Desvantagens: | - Falta de conteúdo exclusivo  - Sem aplicação móvel  - Pouca atenção ao design |
| O que falta: | - Aplicação móvel  - Sistema de pontos |

Fonte: Elaboração Própria

# Desenho

Antes do início do desenvolvimento da aplicação, de modo a obter uma certa ideia relativamente ao que queríamos fazer e para ir em encontro com todos os requisitos da UC foram definidas as seguintes *mockups* e modelos de dados.

## *Mockups*

Para o nosso projeto, desde início, efetuamos vários desenhos e esboços para que tudo se fosse realizar de forma planeada e também para que a interface das nossas plataformas fosse ter uma aparência única e simples.

Incluído nas nossas *mockup’s* temos a Aplicação Móvel, Website (*Frontend* e *Backend*) ([Anexo C](#_Anexo_C)), ([Anexo D](#_Anexo_D)), ([Anexo E](#_Anexo_E)), ([Anexo F](#_Anexo_F)).

## Modelo de dados

No modelo de dados realizamos, num modo inicial, um esboço com todas as tabelas e campos que iriam existir, para que através do mesmo nos fosse mais simples iniciar a introdução de dados. Poderemos ver o mesmo no ([Anexo A](#_Anexo_B)).

# Implementação

## Conceitos e Tecnologias utilizados

Durante a implementação deste projeto foram utilizados diversos conceitos e tecnologias lecionados durante o curso. Sendo que através dos mesmo foi-nos possível obter um resultado final que permita o tratamento de dados, de modo a produzir informação útil ao utilizador.

Tabela 8 - Destaque de algumas tecnologias e conceitos utilizados

|  |  |
| --- | --- |
| **Tecnologia** | **Conceitos** |
| PHP | MVC (Model View Controller) |
| Java | Singleton |
| SQL | Scrum |
| HTML | Frontend |
| CSS | Backend |
| JavaScript | RESTfull API |
| GitHub/GitKraken | Figma |
| Yii2 |  |
| Wamp |  |
| Glide |  |
| Volley |  |
| SQLite |  |
| MapBox |  |
| RBAC |  |

Fonte: Elaboração Própria

## Base de dados

Após a escolha do nosso tema, a primeira coisa a ser realizada foi o esboço da base de dados, com as respetivas tabelas e campos onde viriam a ser armazenadas todas as informações da nossa aplicação. ([Anexo A](#_Anexos))

De seguida, iniciamos a conversão do esboço para uma base de dados, tendo sido utilizado o MYSQL.

No ([Anexo G](#_Anexo_G)) podemos ver o resultado final da base de dados do sistema de GFast.

## Aplicação *Web*

A aplicação *WEB* foi realizada tendo como base a linguagem de programação PHP, juntamente com uma *FrameWorkI,* denominada de Yii2.

Istotraz diversas vantagens pois elimina a necessidade de realizar tarefas base, tais como a criação de *Backend*, *Frontend* e o respetivo registo/login mais seguro. Posto isto, não só conduz a uma maior eficiência de tempo, como também contribui para uma maior segurança da aplicação.

Por outro, fornece diversos ficheiros-exemplo, que poderão vir a ajudar o programador a aprender a utilizar o Yii2.

## MVC

O Yii2 utiliza uma arquitetura MVC (Modelo – Vista – Controlador), para uma maior organização da aplicação e lógica de disposição de ficheiros, arquitetura essa que esteve sempre presente na realização do projeto.

O **Modelo** visa servir de estrutura de determinada tabela da base de dados, sendo lá onde se irá gerir a informação vinda da mesma.

A **Vista** é onde os dados são apresentados ao utilizador, já processados e de forma mais elegante. Porém, para além da linguagem de programação PHP é ainda utilizado HTML, CSS e JavaScript.

O **Controlador**, por sua vez, efetua a gestão dos dois mencionados anteriormente, ao receber os dados vindos do Modelo, processando-os, e, de seguida, enviando-os para a Vista.

## GII

Apesar da arquitetura acima mencionada já vir implementada, ao gerar o novo projeto Yii2 através de pastas organizadas torna-se necessário criar os respetivos ficheiros.

Para tal, foi utilizada uma funcionalidade chamada GII, que permite ao programador gerar os respetivos modelos das tabelas da base de dados, seus controladores e também as vistas CRUD.

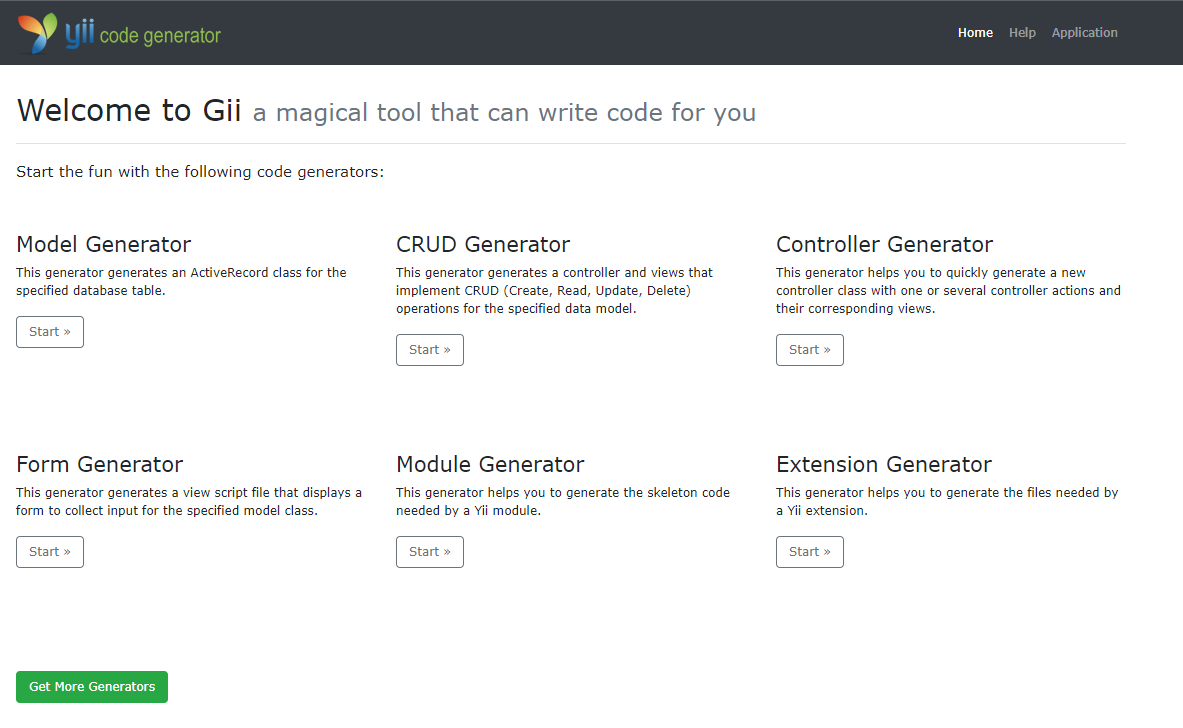


Figura 13- Menu Inicial de GII

Fonte: Dashboard Gii

## ACF e RBAC

Para proteger o website foram utilizados o RBAC e ACF, que são métodos de autorização provenientes do Yii2, sendo que ambos consistem num processo que verifica se o utilizador tem permissões suficientes para a realização de uma determinada tarefa.

Inicialmente, foi desenhado um esboço que atribui aos 4 *roles* definidos (Admin, Gestor, Funcionário e Cliente) as permissões necessárias para aceder a determinados locais do *website*.

Figura 14 - Esboço das permissões definida para cada Role

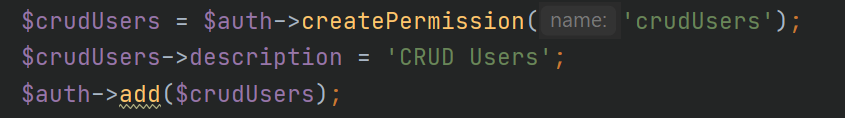


Fonte: Elaboração Própria

De seguida, todas as permissões e *roles* foram geradas e armazenadas na base de dados através de *scaffolding.* Na Figura 15 está presente um exemplo da criação da permissão “crudUsers”.

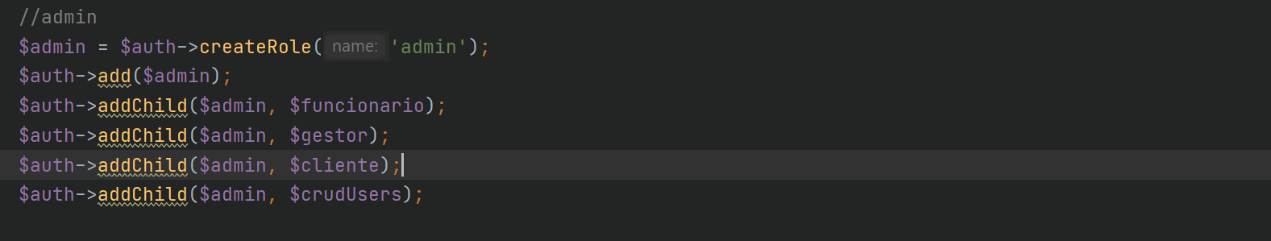
Figura 15 - Exemplo da criação de uma permissão chamada “crudUsers”

Fonte: Elaboração Própria



A Figura 16 apresenta um exemplo de criação de uma *Role*, denominada de “admin”, que tem a permissão de fazer tudo o que o cliente, o funcionário e o gestor fazem, com a adição de também ter a permissão “crudUsers”.

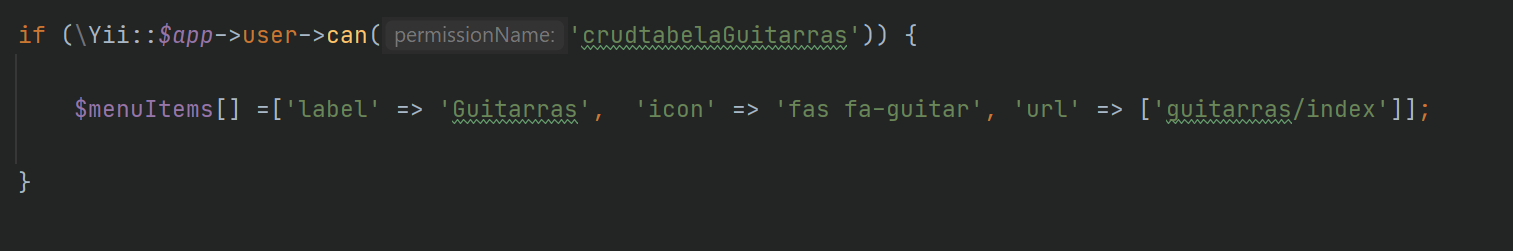
Figura 16 - Exemplo da criação do Role admin



Fonte: Elaboração Própria

Uma vez criadas, uma das principais formas de invocar as *roles* e as permissões é através de um if(), que compara uma permissão com o *role* do utilizador logado.

Figura 17- Exemplo de implementação do RBAC



Fonte: Elaboração Própria

Outra forma de invocar as permissões é através do ACF, que é definido no controlador e consiste num método de autorização de acesso às rotas do url. Deste modo, através desta ferramenta, o acesso a determinadas rotas do url pode estar bloqueado a certos utilizadores, levando a que apenas alguns possam acedê-las.

No exemplo abaixo, o utilizador “admin” é o único que tem permissões para aceder a qualquer uma das rotas do controlador em questão.



Figura 18 - Exemplo de ACF

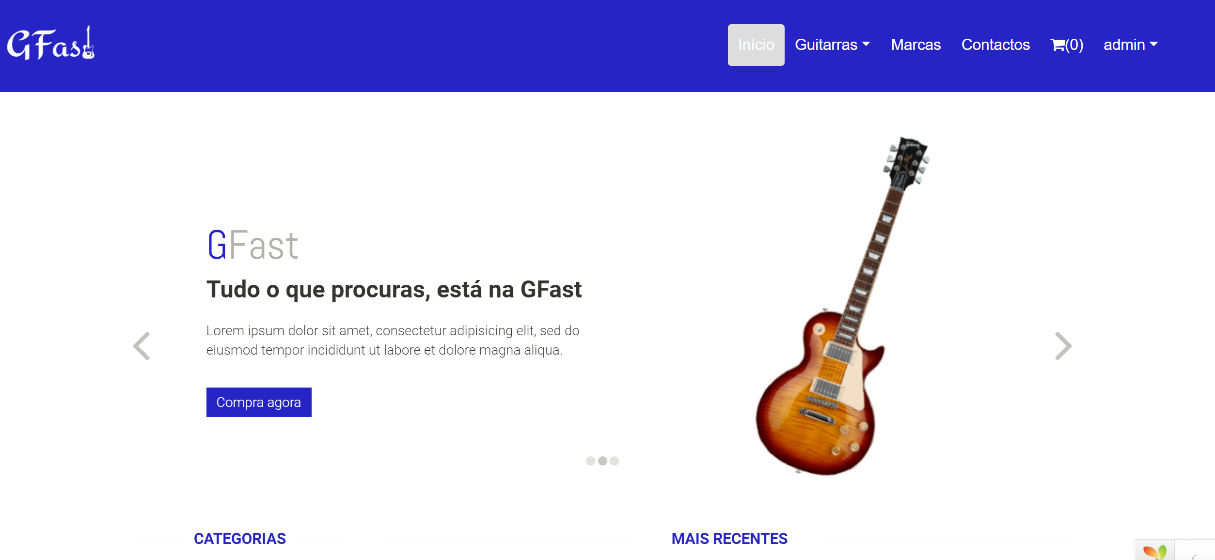
Fonte: Elaboração Própria

## Templates

Após a criação dos métodos de autenticação, tanto no *Backend* como no *Frontend*, para a adenda da Etapa 1 do projeto foi necessário modificá-lo visualmente, de maneira a ficar mais apelativo ao utilizador. Para tal, foram adquiridos dois *templates* que conjugam Javascript, HTML e CSS.

No Frontend é utilizado um *template* chamado “E Shopper Free Website Template”.

Figura 19 - Template Frontend

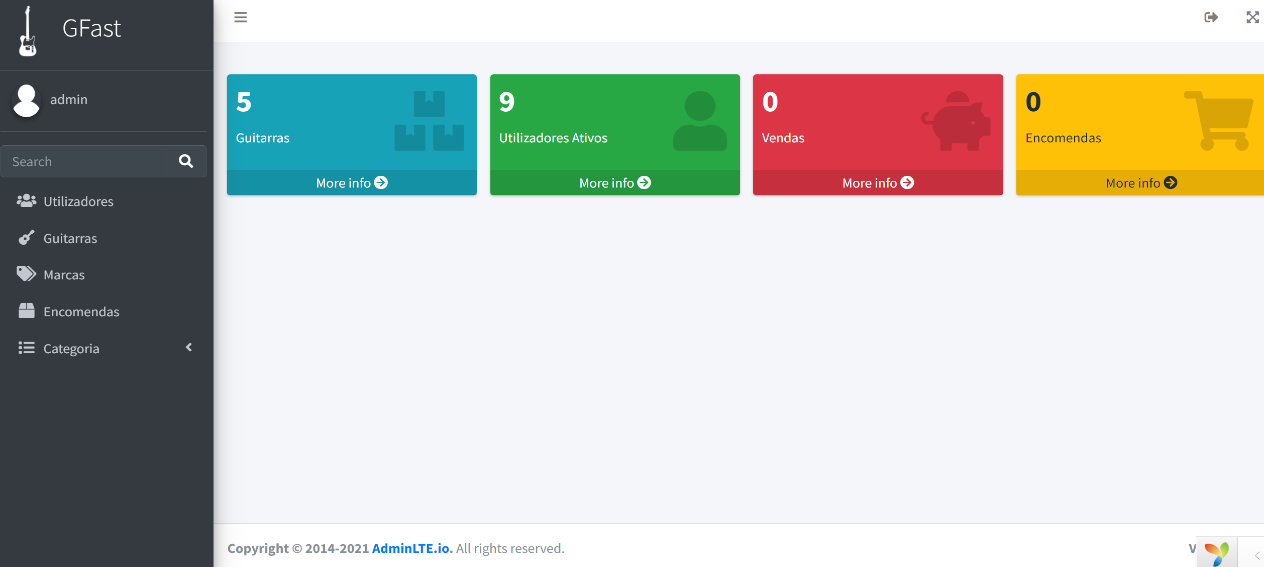


Fonte: Elaboração Própria

No Backend é utilizado um *template* denominado de “Admin LTE 3”, criado pelo utilizador “hail” em 25 de abril de 2020, sendo que foi adquirido através da página de extensões do Yii2.

*Figura 20 - Template Backend*

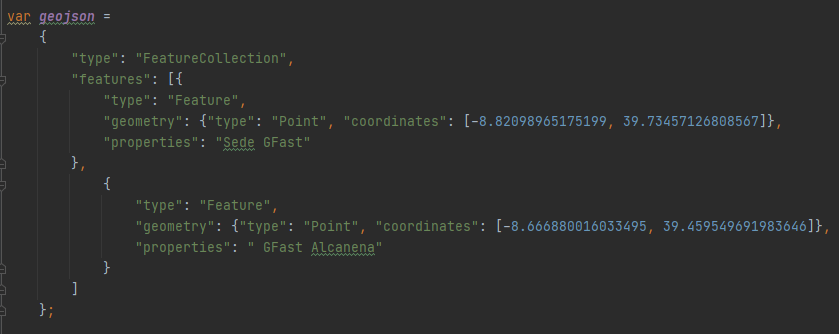
Fonte: Elaboração Própria com base na página de extensões do Yii2



## EXTRAS

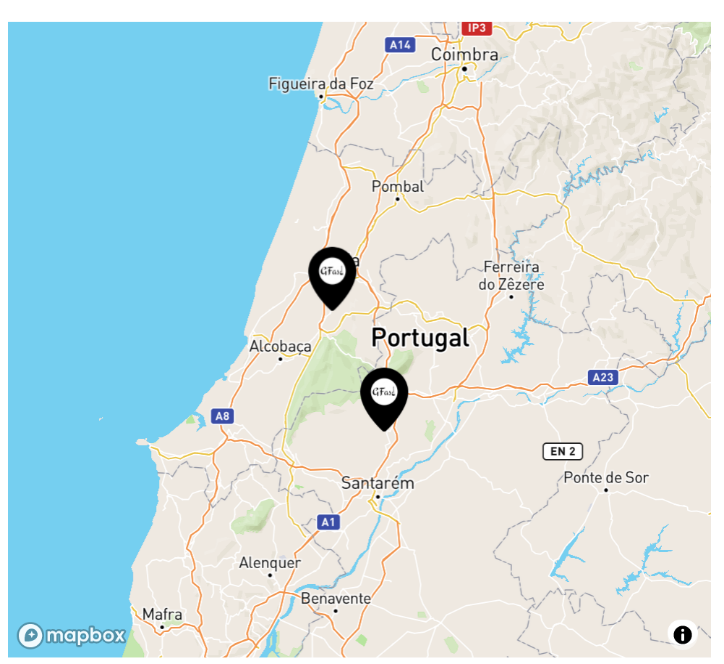
Na página de contato foi criado um mapa como extra com as diversas localizações fictícias das lojas físicas da “GFast”. Para a criação deste mapa é chamada uma API, denominada mapbox, que cria o mapa e, através de coordenadas GPS, cria pontos no mapa.

Figura 21 - Criação de pontos no mapa através de Coordenadas



Fonte: Elaboração Própria

Figura 22 - Mapa da localização de lojas



Fonte: Elaboração Própria

* 1. **ANDROID APLICAÇÃO MÓVEL**

A aplicação móvel também tem implementada uma arquitetura MVC, que a organiza e facilita o processo de construção da mesma.

Para a criação desta aplicação é utilizado diversos *packages* tais como:

Tabela 9 - Packages da aplicação móvel

|  |  |  |
| --- | --- | --- |
| **Adaptadores** | É onde se armazena os ficheiros que iram popular as “listviews” das vistas com as informações que vêm da API ou da base de dados local (SQLite) |  |
| **Listeners** | É onde estão armazenados os ficheiros que estão à escuta para fazer o *callback* onde estiverem a ser chamados |  |
| **Modelos** | Está armazenado:  -Modelos da base de dados;  -Singleton  - O ficheiro que faz toda a gestão da base de dados local (GfastBDHelper);  - MosquittoCallBack |  |
| **Utils** | É onde está armazenado o ficheiro que converte as respostas JSON vindas da API para objetos |  |
| **Vistas** | Onde estão todas as atividades e fragmentos que são responsáveis pela interface da aplicação |  |

Fonte: Elaboração Própria

## Singleton

O Singleton é o ficheiro que faz uma gestão global de dados, tanto da base de dados local como dos pedidos à API.

Os pedidos à API que foram utilizados são os seguintes:

* **GET**
* Obter todas as guitarras na base de dados
* Obter lista de favoritos, através do Id de utilizador
* **POST**
* Realizar Login na Aplicação
* Registar um utilizador
* Adicionar guitarra aos favoritos
* **PUT**
* Editar Utilizador
* **DELETE**
* Eliminar guitarra dos favoritos

Para os pedidos é utilizada a biblioteca *volley*, que recebe como resposta dados vindos da API e, posteriormente, converte a resposta JSON num objeto através do *GFastJsonParser* que tem como objetivo esse mesmo efeito. De seguida, insere esse objeto na base de dados local e chama um *callback* para atualizar dados em tempo real.

Caso a resposta seja um erro, a biblioteca *volley* dá a possibilidade ao programador de implementar código para que a aplicação não se interrompa.

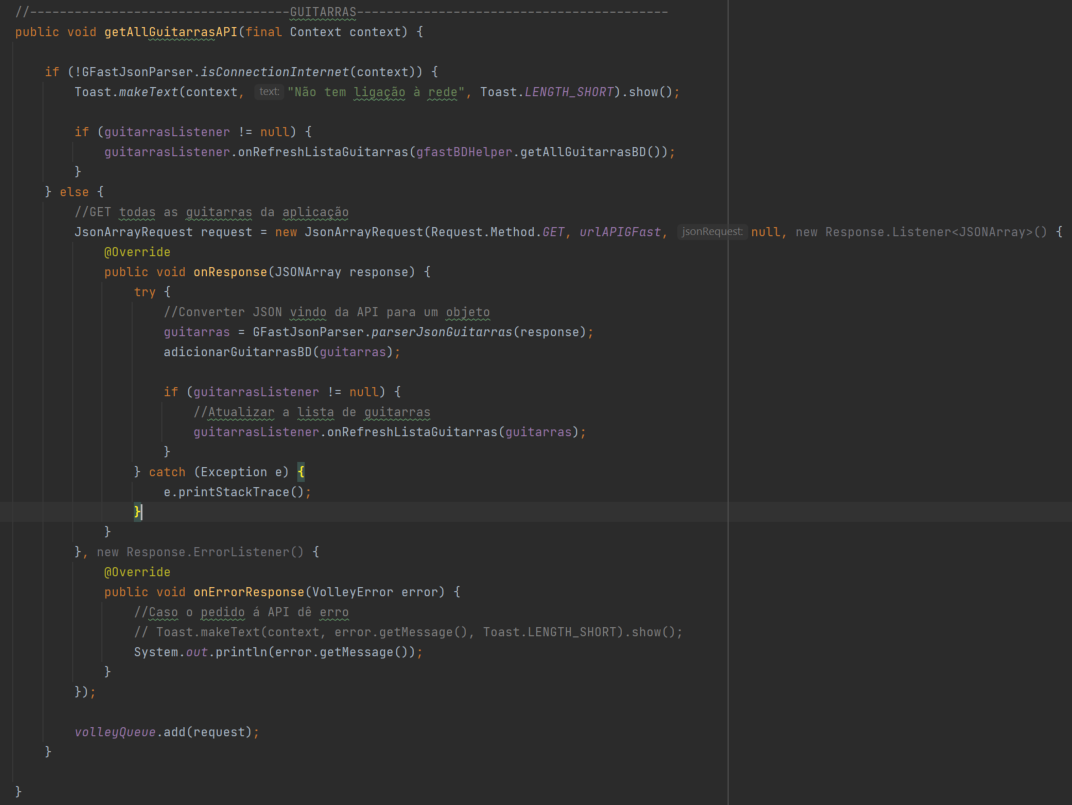


Figura 23 - Exemplo de Pedido Get utilizando o Volley

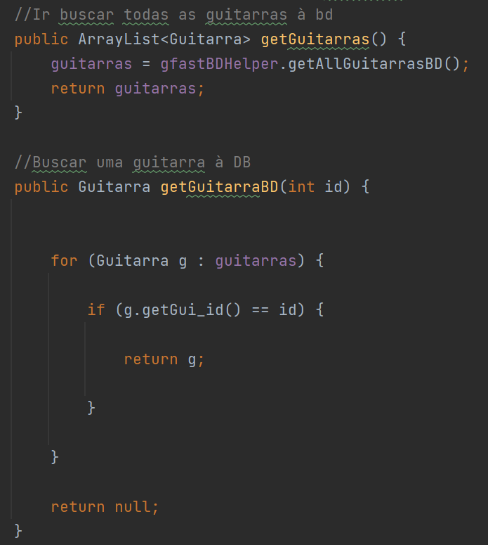
Fonte: Elaboração Própria

Para além dos pedidos à API, o Singleton, em comunicação com o ficheiro “GfastBDhelper”, faz a gestão da base de local, que tem como objetivos:

* Permitir a utilização da aplicação sem internet, pois assim que é realizado um pedido os dados recebidos serão armazenados;
* Armazenar dados de login do utilizador;

O exemplo abaixo mostra uma função que vai buscar todas as guitarras armazenadas na base de dados local, e uma outra função que busca uma guitarra pelo seu id.

Figura 24 Exemplo de gestão da bd local



Fonte: Elaboração Própria

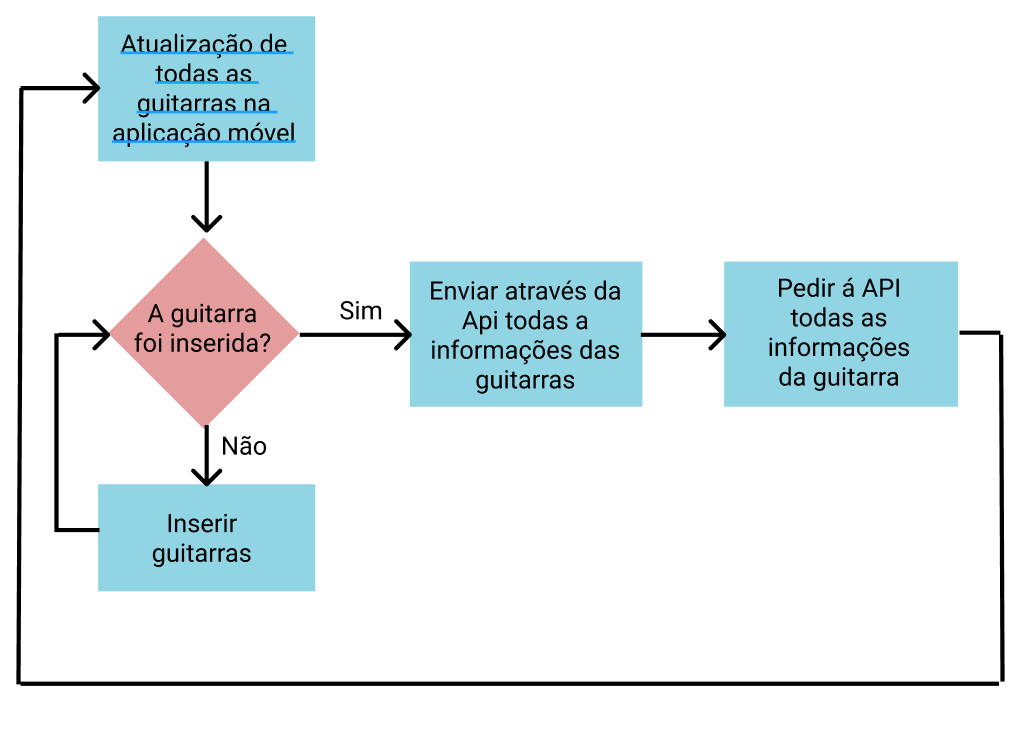
## Atividades e Fragmentos

* **MenuMainActivity**
* Atividade principal que contém o menu lateral com acesso aos fragmentos.
* **LoginActivity**
* Atividade que permite ao utilizador realizar login na aplicação; caso este utilizador já tenha feito login anteriormente, esse login estará guardado na base de dados e permitirá entrar na aplicação sem ter que passar por esta atividade.
* **RegisterActivity**
* Atividade que permite ao utilizador fazer o registo na aplicação; ao inserir os seus dados estes serão enviados para API através de um pedido GET, que, por sua vez, os irão guardar na base de dados.
* **DetalhesGuitarraActivity**
* Informações acerca de uma guitarra em questão, tais como marca, modelo, preço, descrição e foto que é obtida através da biblioteca Glide.
* **ListaGuitarrasFragment**
* Lista de todas as guitarras vindas da API através de um pedido GET ou, caso não haja conexão a uma rede de internet, aparecem as que estão armazenadas na base de dados local.
* Nesta lista é possível abrir os detalhes de cada guitarra, apenas ao pressionar nelas.
* Existe uma barra de pesquisa para procurar as guitarras pelo nome.
* **PerfilFragment**
* É possível ao abrir apresentar todos os dados do utilizador, e, caso pretender, alterá-los através de um pedido PUT
* **FavoritosFragment**
* Lista de todas as guitarras que estão nos favoritos vindas da API através de um pedido GET ou, caso não haja conexão à uma rede de internet, aparecem as que estão armazenadas na base de dados local.
* Nesta lista é possível ao pressionar uma guitarra abrir os seus detalhes.
  1. **API**

Na Figura 25 podemos verificar a implementação realizada através da API, onde se verifica se é ou não realizada uma inserção de guitarras por parte do gestor ou administrador.

De seguida, são enviadas todas as informações relacionadas com os campos preenchidos (especificações) e, através da API, é realizada uma ponte da plataforma para a aplicação, na qual é realizada uma atualização de guitarras.

Figura 25 – implementação da API

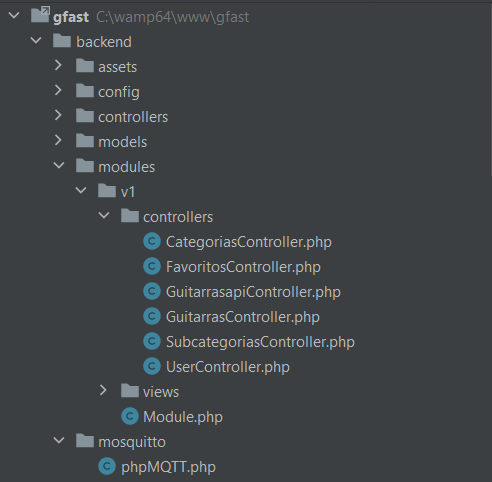


Fonte: Elaboração Própria

Todos os seguintes resultados são gerados através de controladores criados na pasta de *backend* do website:

Figura 26 – Pasta de controladores

Fonte: Elaboração Própria



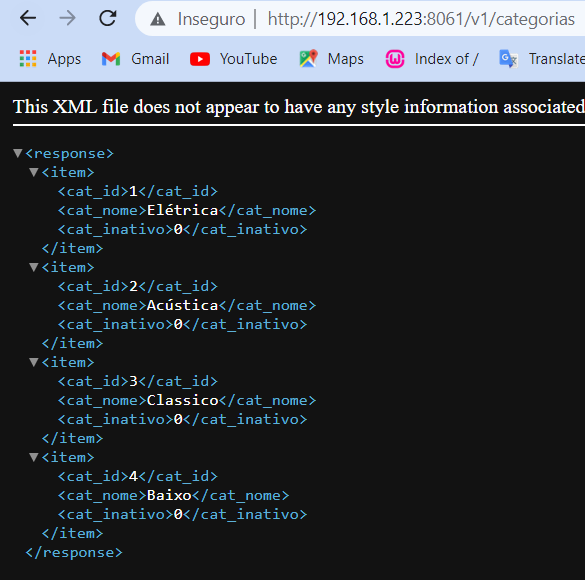
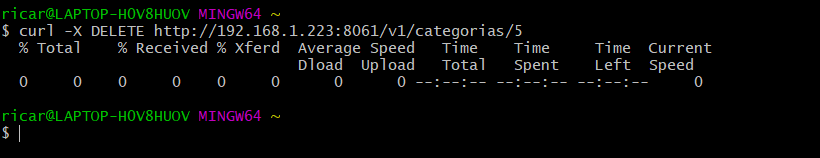
# cURL resultado no CMD

Nas seguintes imagens podemos verificar um exemplo de GET, POST, PUT e DELETE de categorias que é resultado no CMD

* + - 1. **Categorias**

GET – Categorias

*Figura 27 – Exemplo GET Categorias*



Fonte: Elaboração Própria

POST – Categorias

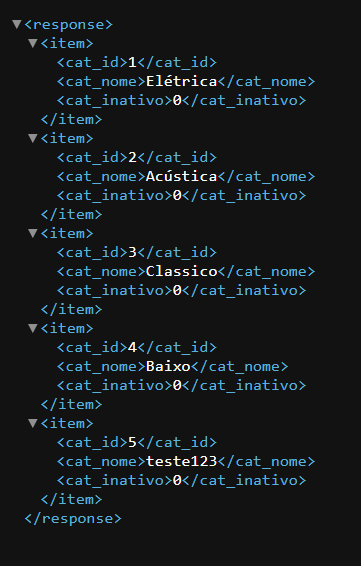
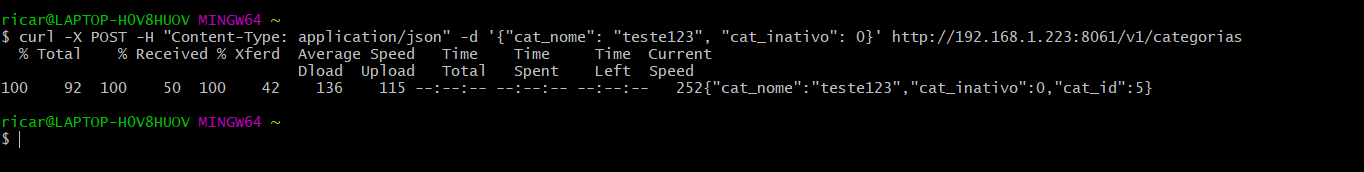


Figura 28 – Exemplo POST Categorias

Fonte: Elaboração Própria

PUT – Categorias

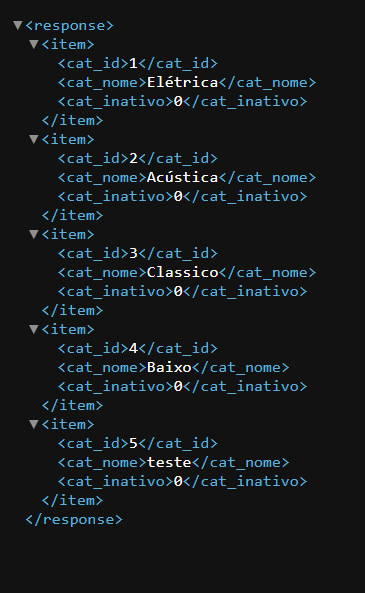
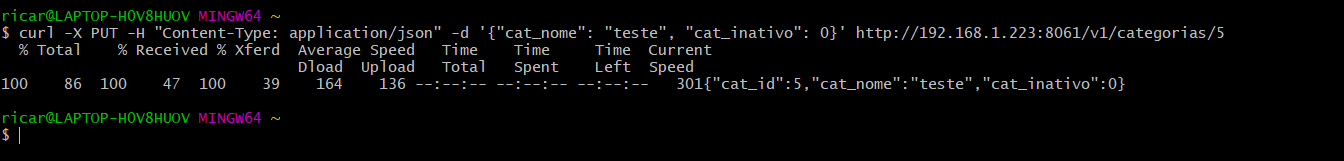


Figura 29 – Exemplo PUT Categorias

Fonte: Elaboração Própria

DELETE – Categorias

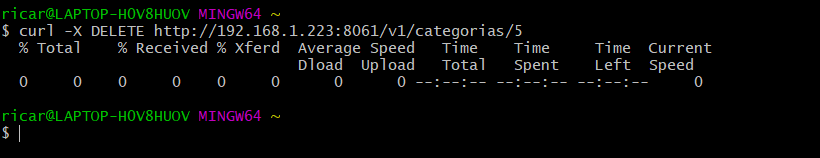
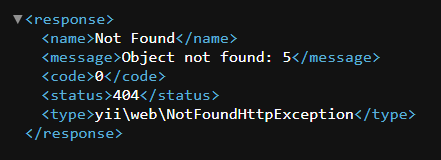


Figura 30 – Exemplo DELETE Categorias

Fonte: Elaboração Própria

# EndPoints

Todos os *endpoints* devem conter no seu caminho /ip:8061/….

Por exemplo o *endpoint* para obter o utilizador com o id: - <http://ip:8061/users/1>

### **Users**

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descrição |
| GET | ip/v1/users | Ver todos os utilizadores |
| POST | ip/v1/users/1 | Ver cada utilizador |

### **Guitarras**

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descrição |
| GET | ip/v1/guitarrasapi | Todas as guitarras |
| POST | ip/v1/guitarrasapis/1 | Ver cada guitarra |

### **Subcategorias**

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descrição |
| GET | ip/v1/subcategorias/1/guitarras | Ver guitarras de cada subcategoria |
| GET | ip/v1/subcategorias | Ver todas as subcategorias |

### **Categorias**

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descrição |
| GET | ip/v1/categorias/1/subcategorias | Ver subcategorias de cada categoria |
| GET | ip/gfast/v1/categorias | Ver todas as categorias |

* + 1. **Messaging**

No projeto pretende-se implementar a componente de messaging (mqtt) na atualização de guitarras em Android, uma vez que todas as guitarras inseridas por um administrador ou gestor na plataforma Gfast irão automaticamente atualizar na App.

Para a realização da atualização foi utilizado Andoid Studio e Php Storm.

Através do controlador “Guitarras.php”, na pasta “Common”, são passadas todas as informações acerca das guitarras inseridas, como poderemos verificar nas seguintes imagens:

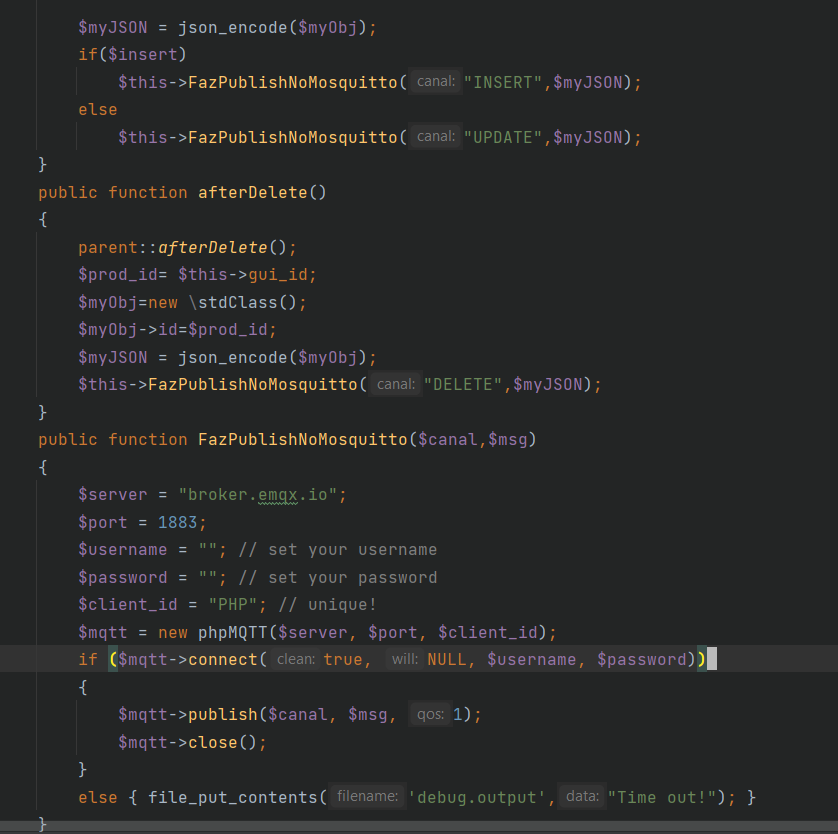
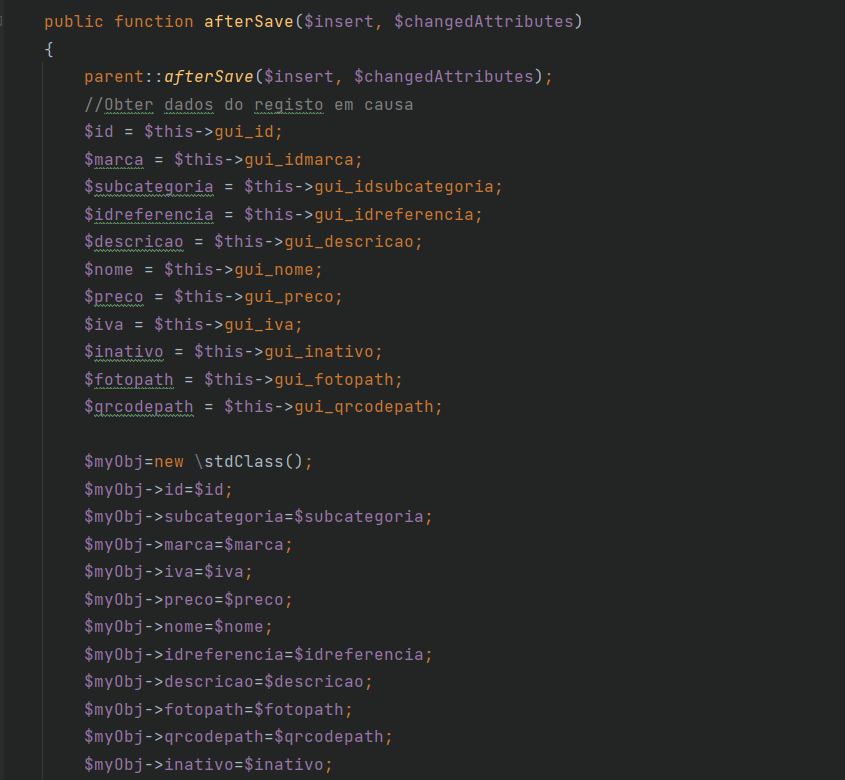


Figura 31 – Código de messaging

Fonte: Elaboração Própria

# Testes

## BACKEND

Testes unitários e objetivos

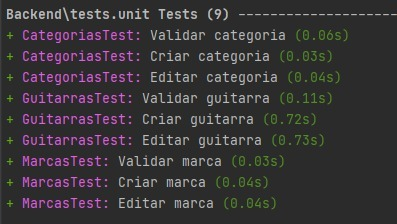
Tabela 10 – Backend: Testes unitários

|  |  |
| --- | --- |
|  | CategoriasTest |
| 1 | Nome: testValidarCategoria  Objetivo: Teste de validação para categorias, para verificar o funcionamento. |
| 2 | Nome: testCriarCategoria  Objetivo: Teste para verificar a inserção de categorias. |
| 3 | Nome: testEditarCategoria  Objetivo: Teste para editar categorias. |
|  | **GuitarrasTest** |
| 4 | Nome: testValidarGuitarra  Objetivo: Teste de validação para guitarras, de modo a verificar o funcionamento. |
| 5 | Nome: testCriarGuitarra  Objetivo: Teste para verificar a inserção de guitarras. |
| 6 | Nome: testEditarGuitarra  Objetivo: Teste para editar guitarras. |
|  | **MarcasTest** |
| 7 | Nome: testValidarMarca  Objetivo: Teste de validação para marcas, para verificar o funcionamento. |
| 8 | Nome: testCriarMarca  Objetivo: Teste para verificar a inserção de marcas. |
| 9 | Nome: testEditarMarca  Objetivo: Teste para editar marcas. |

Fonte: Elaboração Própria

Figura 32 – testes unitários backend

Fonte: Elaboração Própria



Testes Funcionais e objetivos

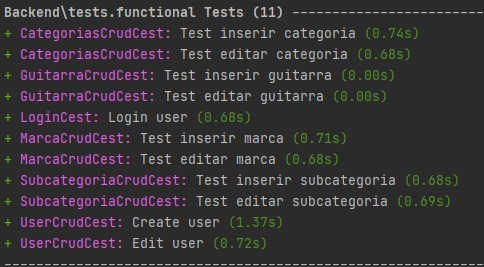
Tabela 11 – Backend: Testes Funcionais

|  |  |
| --- | --- |
|  | CategoriasCrudCest |
| 1 | Nome: testInserirCategoria  Objetivo: Teste para verificar a inserção de categorias. |
| 2 | Nome: testEditarCategoria  Objetivo: Teste para editar categorias. |
|  | **GuitarraCrudCest** |
| 3 | Nome: testInserirGuitarra  Objetivo: Teste para verificar a inserção de guitarras. |
| 4 | Nome: testEditarGuitarra  Objetivo: Teste para editar guitarras. |
|  | **LoginCest** |
| 5 | Nome: loginUser  Objetivo: Teste para realizar login a um utilizador. |
|  | **MarcaCrudCest** |
| 6 | Nome: testInserirMarca  Objetivo: Teste para verificar a inserção de marca. |
| 7 | Nome: testEditarMarca  Objetivo: Teste para editar marcas. |
|  | **SubcategoriaCrudCest** |
| 8 | Nome: testInserirSubcategoria  Objetivo: Teste para verificar a inserção de subcategorias. |
| 9 | Nome: testEditarSubcategoria  Objetivo: Teste para editar subcategorias. |
|  | **UserCrudCest** |
| 10 | Nome: createUser  Objetivo: Teste para verificar a criação de utilizador. |
| 11 | Nome: editUser  Objetivo: Teste para editar users. |

Fonte: Elaboração Própria

Figura 33 – testes funcionais backend

Fonte: Elaboração Própria



## FRONTEND

Testes unitários e objetivos

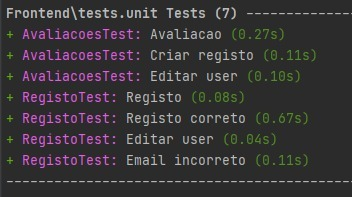
Tabela 12 – Frontend: Testes unitários

|  |  |
| --- | --- |
|  | AvaliacoesTest |
| 1 | Nome: testAvaliacao  Objetivo: Teste para validar as avaliações. |
| 2 | Nome: testCriarRegisto  Objetivo: Teste para criar as avaliações com um utilizador. |
| 3 | Nome: testEditarUser  Objetivo: Teste para editar a avaliação de um utilizador. |
|  | **RegistoTest** |
| 4 | Nome: testRegisto  Objetivo: Teste para registar um utilizador incorretamente. |
| 5 | Nome: testRegistoCorreto  Objetivo: Teste para registar um utilizador corretamente. |
| 6 | Nome: testEditarUser  Objetivo: Teste para editar um utilizador. |
| 7 | Nome: testEmailIncorreto  Objetivo: Teste para registar mal o email a um utilizador. |

Fonte: Elaboração Própria

Figura 34 – Testes unitários frontend

Fonte: Elaboração Própria



Testes Funcionais e objetivos

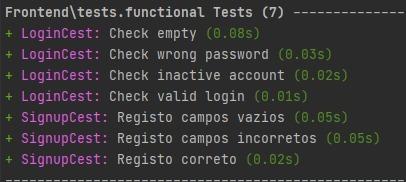
Tabela 13 – Frontend: Testes Funcionais

|  |  |
| --- | --- |
|  | LoginCest |
| 1 | Nome: checkEmpty  Objetivo: Teste para verificar se os campos do login estão vazios. |
| 2 | Nome: checkWrongPassword  Objetivo: Teste para verificar se a password está incorreta. |
| 3 | Nome: checkInactiveAccount  Objetivo: Teste para ver se um utilizador está inativo. |
| 4 | Nome: checkValidLogin  Objetivo: Teste para realizar um login correto. |
|  | **SignupCest** |
| 5 | Nome: registoCamposVazios  Objetivo: Teste para verificar se os campos do registo estão vazios. |
| 6 | Nome: registoCamposIncorretos  Objetivo: Teste para verificar se os campos estão incorretos. |
| 7 | Nome: registoCorreto  Objetivo: Teste para realizar um registo correto. |

Fonte: Elaboração Própria

Figura 35 – Testes funcionais backend

Fonte: Elaboração Própria



Testes de Aceitação e objetivos

Tabela 14 – Frontend: Testes de Aceitação

|  |  |
| --- | --- |
|  | AvaliacaoCest |
| 1 | Nome: CRUDAvaliacao  Objetivo: Teste para inserir, editar e apagar uma avaliação de um utilizador. |

Fonte: Elaboração Própria

Figura 36 – Teste complete de aceitação



Fonte: Elaboração Própria

# Conclusão e trabalho futuro

De modo a concluir este projeto, é de mencionar que a sua realização foi muito mais pertinente do que o esperado, sendo que adquirimos bastante conhecimento ao longo do seu desenvolvimento.

Durante a sua elaboração não forma encontradas dificuldades consideradas graves, visto que este foi realizado de forma organizada, através de observações diárias e pesquisas na internet acerca das implementações pretendidas.

No fim de tudo, podemos com toda a certeza concluir que os objetivos planeados foram todos completados, e as respetivas tarefas foram sujeitas a uma divisão que beneficiou o projeto.

No que se refere às tarefas, estas foram efetuadas por cada membro, sendo que houve um grande apoio entre os mesmos, através de dicas e explicações nos assuntos em que cada tinha mais dificuldades.

Por fim, este projeto forneceu um grande conhecimento, preparando-nos não só para projetos futuros, assim como também para o mundo de trabalho.

# Bibliografia

ESP. (n.d.). *ESP*. Retrieved October 9, 2021, from https://www.espguitars.com/

Ludimusic. (2022). *Ludimusic*. https://www.ludimusic.com/?gclid=Cj0KCQiApL2QBhC8ARIsAGMm-KE5e3-Gm8mZ3QMkWX\_-S050cPV7goWgqq3O6ABU0jJqHHB5vmYakz4aAtsaEALw\_wcB

Thomann. (2022). *Thomann*. https://www.thomann.de/gb/index.html

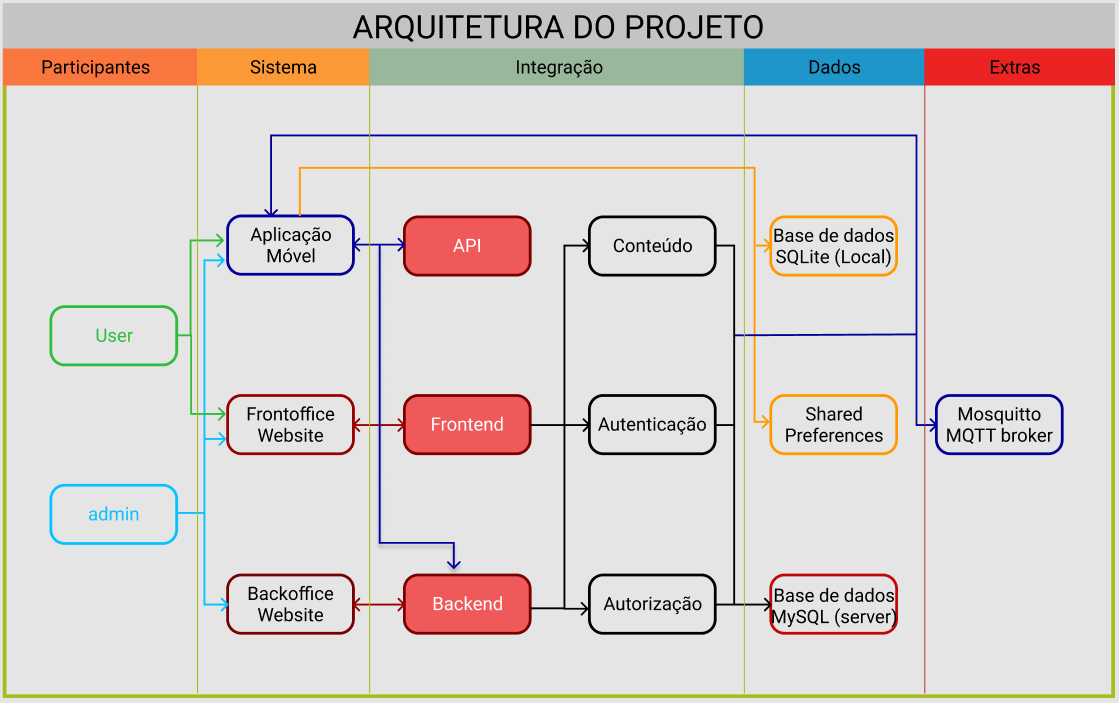
# Anexos

**Acessos ao projeto**

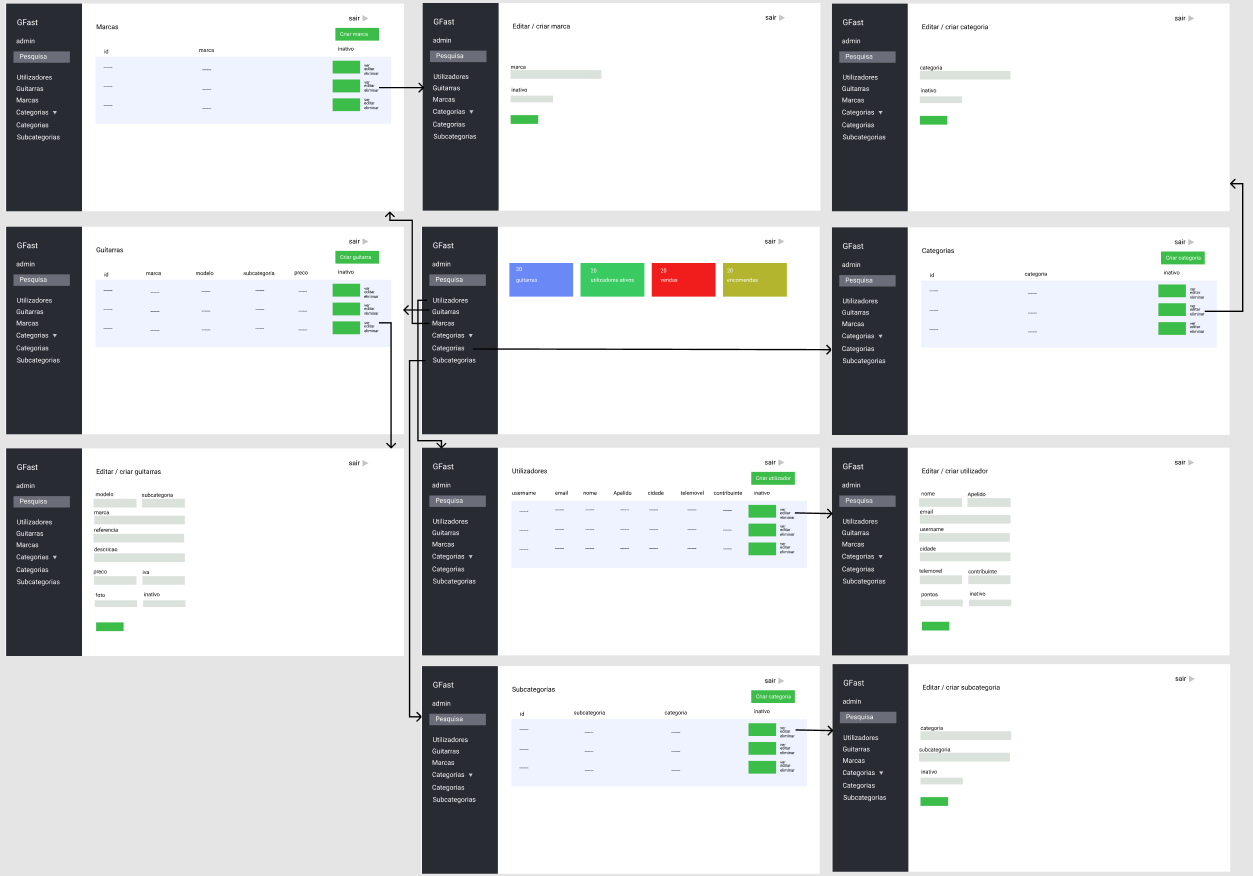
|  |  |
| --- | --- |
| Username | Pasword |
| cliente | 1234567890 |
| func | 1234567890 |
| gestor | 1234567890 |
| admin | admin123 |

# Anexo A

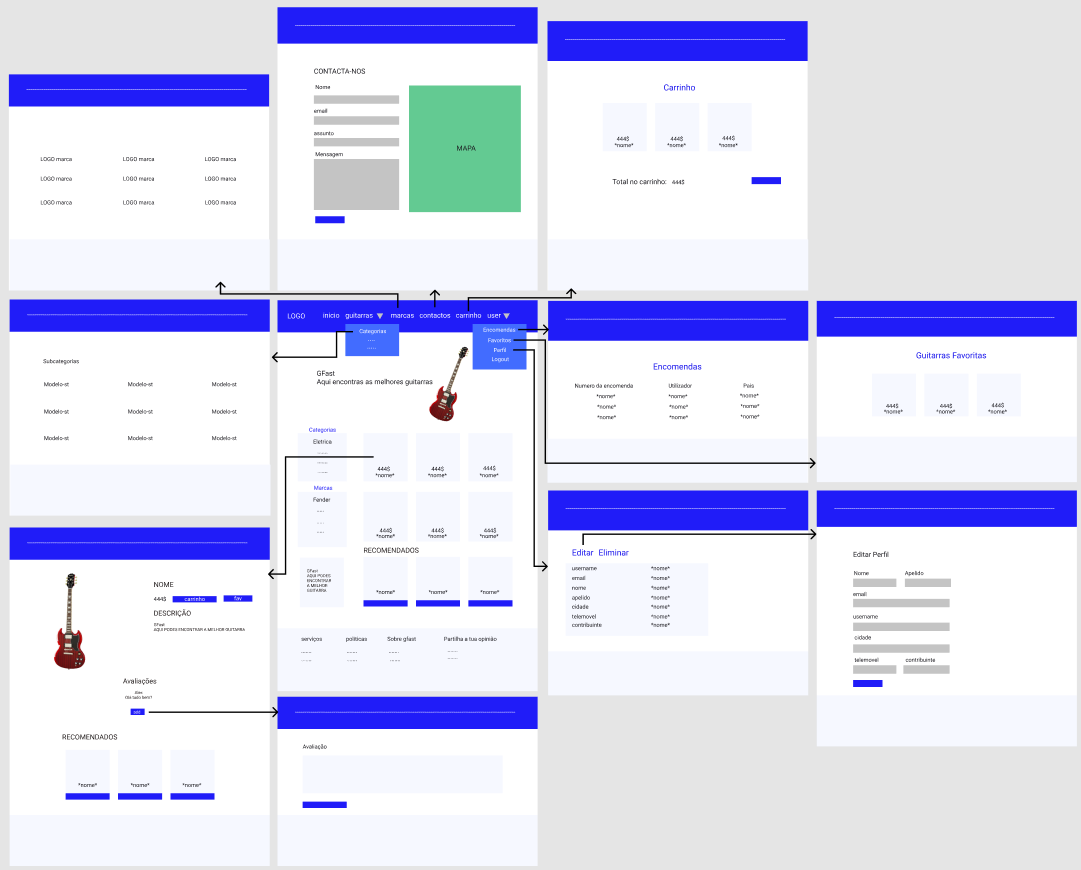
# Anexo B



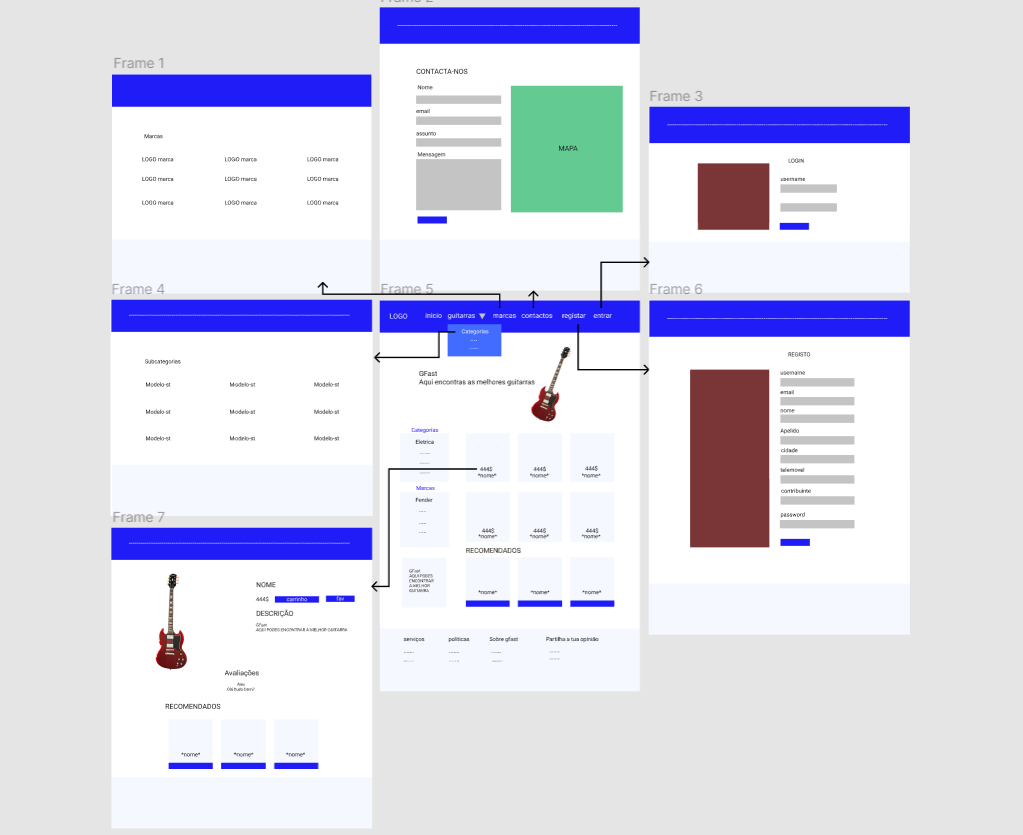
# Anexo C



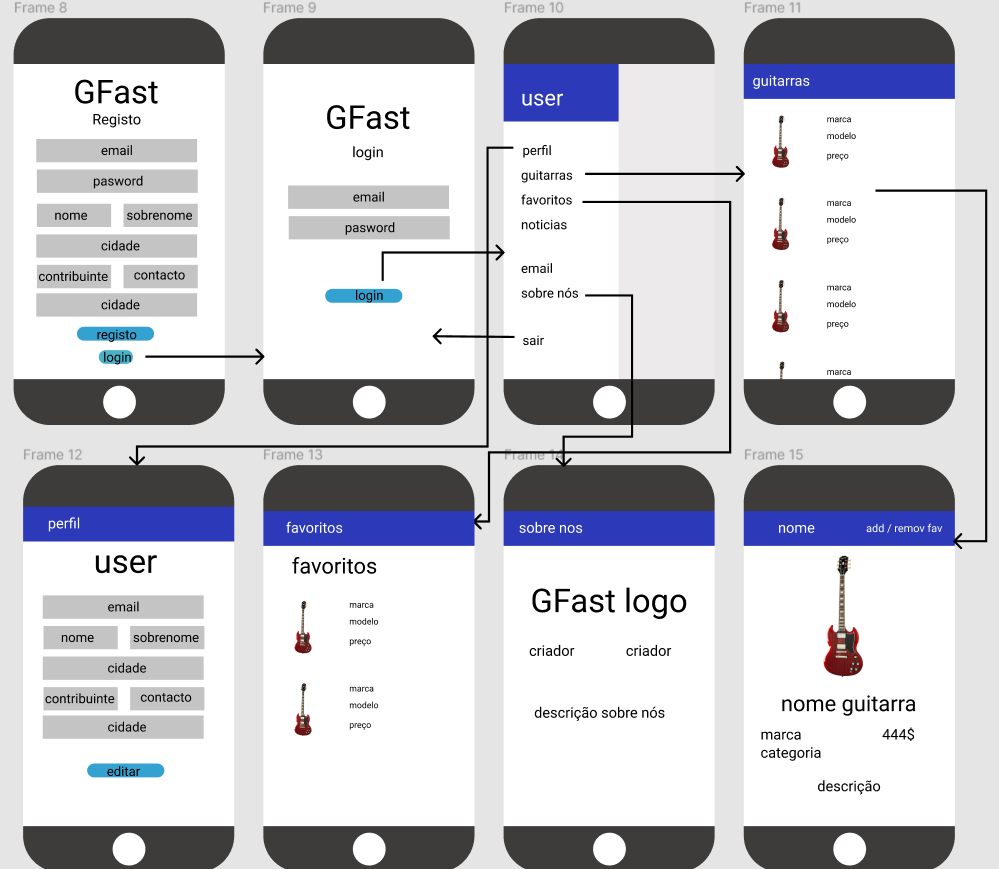
# Anexo D



# Anexo E



# Anexo F



# Anexo G

