



INSTITUTO POLITÉCNICO DE LEIRIA

GFAST

RELATÓRIO DE PROJETO EM SISTEMAS DE
INFORMAÇÃO

ALEXANDRE LEVCHENKO N°2201155
DUARTE FRAZÃO PEREIRA N°2190715

PROGRAMAÇÃO EM SISTEMAS DE
INFORMAÇÃO

18 / 02 / 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 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

<i>Agradecimentos</i>	<i>iv</i>
<i>Resumo</i>	<i>v</i>
<i>Índices</i>	<i>vii</i>
1. Introdução	2
2. Metodologia	4
3. Arquitetura do Sistema	5
4. Gestão do Projeto	6
4.1. Base de dados	6
4.2. Aplicação Móvel	7
4.3. Website	8
4.4. API	9
5. Análise	10
5.1. Tema do Projeto	10
5.2. Objetivos Principais	10
5.3. Requisitos e funcionalidades	10
5.3.1. Website (frontend)	10
5.3.2. Website (backend)	11
5.3.3. Aplicação Móvel	11
5.3.4. API	11
5.4. Análise Concorrencial	12
5.4.1. LudiMusic	12
5.4.2. Thomann	13
5.4.3. ESP	14
6. Desenho	16
6.1. Mockups	16
6.2. Modelo de dados	16
7. Implementação	17
7.1. Conceitos e Tecnologias utilizados	17
7.2. Base de dados	17
7.3. Aplicação Web	18
7.3.1. MVC	18
7.3.2. GII	18
7.3.3. ACF e RBAC	19

7.3.4.	Templates -----	22
7.3.5.	EXTRAS -----	23
7.4.1.	Singleton -----	25
7.4.2.	Atividades e Fragmentos -----	27
7.5.1.	<i>cURL resultado no CMD</i> -----	29
7.5.2.	<i>EndPoints</i> -----	31
1.	Users -----	32
2.	Guitarras -----	32
3.	Subcategorias -----	32
3.1.1.	Categorias -----	32
8.	<i>Testes</i> -----	34
8.1.	BACKEND -----	34
8.2.	FRONTEND -----	36
9.	<i>Conclusão e trabalho futuro</i> -----	40
10.	<i>Bibliografia</i> -----	41
	<i>Anexos</i> -----	1
	<i>Anexo A</i> -----	2
	<i>Anexo B</i> -----	2
	<i>Anexo C</i> -----	4
	<i>Anexo D</i> -----	5
	<i>Anexo E</i> -----	6
	<i>Anexo F</i> -----	7
	<i>Anexo G</i> -----	8

Índice de Tabelas

TABELA 1 - REQUISITOS E FUNCIONALIDADES (FRONTEND)	10
TABELA 2 - REQUISITOS E FUNCIONALIDADES WEB(FRONTEND)	11
TABELA 3 - REQUISITOS E FUNCIONAIS APLICAÇÃO MÓVEL	11
TABELA 4 - REQUISITOS E FUNCIONALIDADES API	11
TABELA 5 - DESCRIÇÃO DA LUDIMUSIC	12
TABELA 6 - DESCRIÇÃO DA THOMANN	13
TABELA 7 - DESCRIÇÃO DA ESP	14
TABELA 8 - DESTAQUE DE ALGUMAS TECNOLOGIAS E CONCEITOS UTILIZADOS	17
TABELA 9 - PACKAGES DA APLICAÇÃO MÓVEL	24
TABELA 10 – BACKEND: TESTES UNITÁRIOS	34
TABELA 11 – BACKEND: TESTES FUNCIONAIS	35
TABELA 12 – FRONTEND: TESTES UNITÁRIOS	36
TABELA 13 – FRONTEND: TESTES FUNCIONAIS	37
TABELA 14 – FRONTEND: TESTES DE ACEITAÇÃO	38

Índice de Figuras

FIGURA 1 – ARQUITETURA DO SISTEMA.....	5
FIGURA 2 – PLANEAMENTO DE TAREFAS: BASE DE DADOS	6
FIGURA 3 – CRONOLOGIA: BASE DE DADOS	6
FIGURA 4 – PLANEAMENTO DE TAREFAS: APLICAÇÃO MÓVEL	7
FIGURA 5 – CRONOLOGIA: APLICAÇÃO MÓVEL	7
FIGURA 6 – PLANEAMENTO DE TAREFAS: PSI (WEB)	8
FIGURA 7 – CRONOLOGIA: PSI (WEB)	8
FIGURA 8 – PLANEAMENTO DE TAREFAS: SIS (API)	9
FIGURA 9 – CRONOLOGIA: SIS (API)	9
FIGURA 10 – WEBSITE DA LUDIMUSIC.....	12
FIGURA 11 – WEBSITE DA THOMANN.....	13
FIGURA 12 – WEBSITE DA ESP	14
FIGURA 13- MENU INICIAL DE GII.....	19
FIGURA 14 - ESBOÇO DAS PERMISSÕES DEFINIDA PARA CADA ROLE	20
FIGURA 15 - EXEMPLO DA CRIAÇÃO DE UMA PERMISSÃO CHAMADA “CRUDUSERS”	20
FIGURA 16 - EXEMPLO DA CRIAÇÃO DO ROLE ADMIN.....	20
FIGURA 17- EXEMPLO DE IMPLEMENTAÇÃO DO RBAC	21
FIGURA 18 - EXEMPLO DE ACF	21
FIGURA 19 - TEMPLATE FRONTEND.....	22
FIGURA 20 - TEMPLATE BACKEND	22
FIGURA 21 - CRIAÇÃO DE PONTOS NO MAPA ATRAVÉS DE COORDENADAS	23
FIGURA 22 - MAPA DA LOCALIZAÇÃO DE LOJAS	23
FIGURA 23 - EXEMPLO DE PEDIDO GET UTILIZANDO O VOLLEY	26
FIGURA 24 EXEMPLO DE GESTÃO DA BD LOCAL	26
FIGURA 25 – IMPLEMENTAÇÃO DA API	28
FIGURA 26 – PASTA DE CONTROLADORES	29
FIGURA 27 – EXEMPLO GET CATEGORIAS	30
FIGURA 28 – EXEMPLO POST CATEGORIAS.....	30
FIGURA 29 – EXEMPLO PUT CATEGORIAS	31
FIGURA 30 – EXEMPLO DELETE CATEGORIAS	31
FIGURA 31 – CÓDIGO DE MESSAGING	33
FIGURA 32 – TESTES UNITÁRIOS BACKEND	35
FIGURA 33 – TESTES FUNCIONAIS BACKEND	36
FIGURA 34 – TESTES UNITÁRIOS FRONTEND.....	37
FIGURA 35 – TESTES FUNCIONAIS BACKEND	38
FIGURA 36 – TESTE COMPLETE DE ACEITAÇÃO	39

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 Markup 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

1. 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.

2. 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 completar 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 tivéssemos 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*.

3. 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.

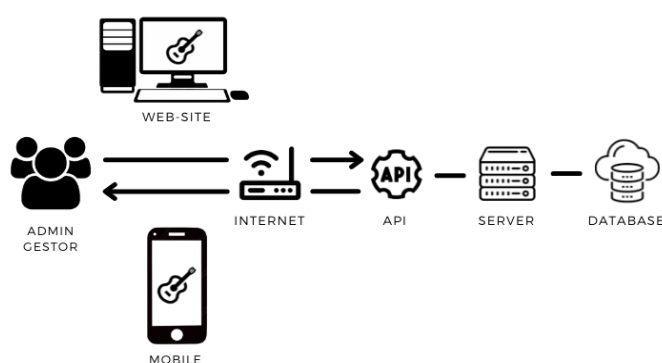
A parte do *Frontend* disponibiliza aos nossos utilizadores uma interface bastante apelativa e com um design bastante simples para facilitar na navegação, enquanto o *Backend* possibilita a parte de administração realizar a gestão de conteúdo inserido no *Frontend*.

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](#) e figura 1)

Figura 1 – Arquitetura do Sistema



Fonte: Elaboração Própria

4. 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.

4.1. BASE DE DADOS

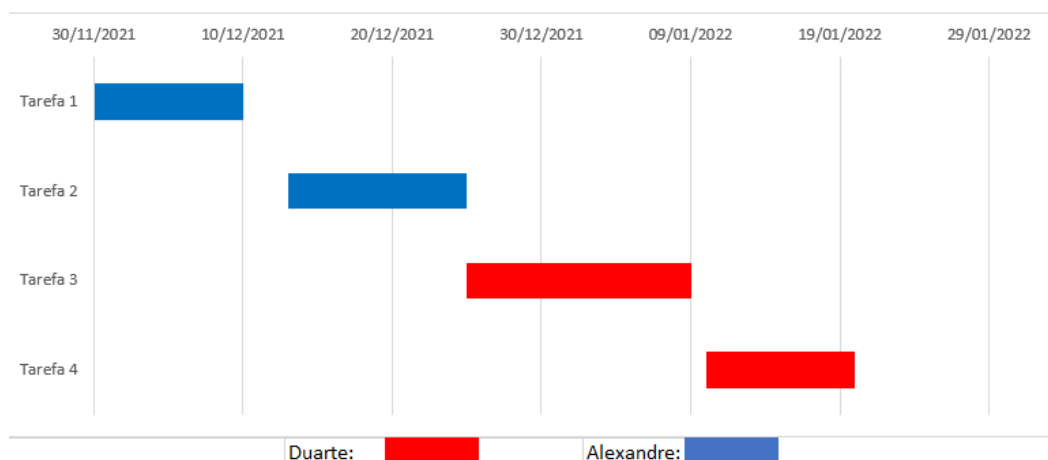
Figura 2 – Planeamento de Tarefas: Base de Dados

Tarefas	Data início	Duração (dias)	Data de conclusão
Tarefa 1	30/11/2021	10	10/12/2021
Tarefa 2	13/12/2021	12	25/12/2021
Tarefa 3	25/12/2021	15	09/01/2022
Tarefa 4	10/01/2022	10	20/01/2022

Tarefas	Designação
Tarefa 1	Desenho de todas as tabelas
Tarefa 2	Criação das tabelas
Tarefa 3	Ligações com chave estrangeira
Tarefa 4	Adicionar tabelas que faltam

Fonte: Elaboração Própria

Figura 3 – Cronologia: Base de Dados



Fonte: Elaboração Própria

4.2. APLICAÇÃO MÓVEL

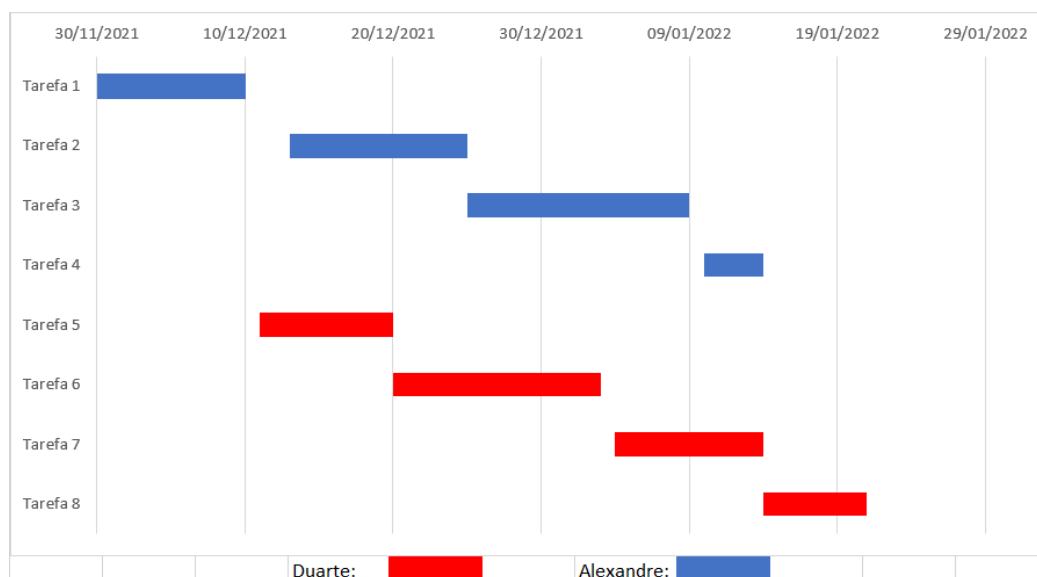
Figura 4 – Planeamento de Tarefas: Aplicação Móvel

Tarefas	Data início	Duração (dias)	Data de conclusão
Tarefa 1	30/11/2021	10	10/12/2021
Tarefa 2	13/12/2021	12	25/12/2021
Tarefa 3	25/12/2021	15	09/01/2022
Tarefa 4	10/01/2022	4	14/01/2022
Tarefa 5	11/12/2021	9	20/12/2021
Tarefa 6	20/12/2021	14	03/01/2022
Tarefa 7	04/01/2022	10	14/01/2022
Tarefa 8	14/01/2022	7	21/01/2022

Tarefas	Designação
Tarefa 1	Layout incluindo as listas e menu lateral
Tarefa 2	Login
Tarefa 3	Registo
Tarefa 4	Logout
Tarefa 5	Editar Perfil
Tarefa 6	Guitarras
Tarefa 7	Favoritos
Tarefa 8	Manter o login de utilizador

Fonte: Elaboração Própria

Figura 5 – Cronologia: Aplicação Móvel



Fonte: Elaboração Própria

4.3. WEBSITE

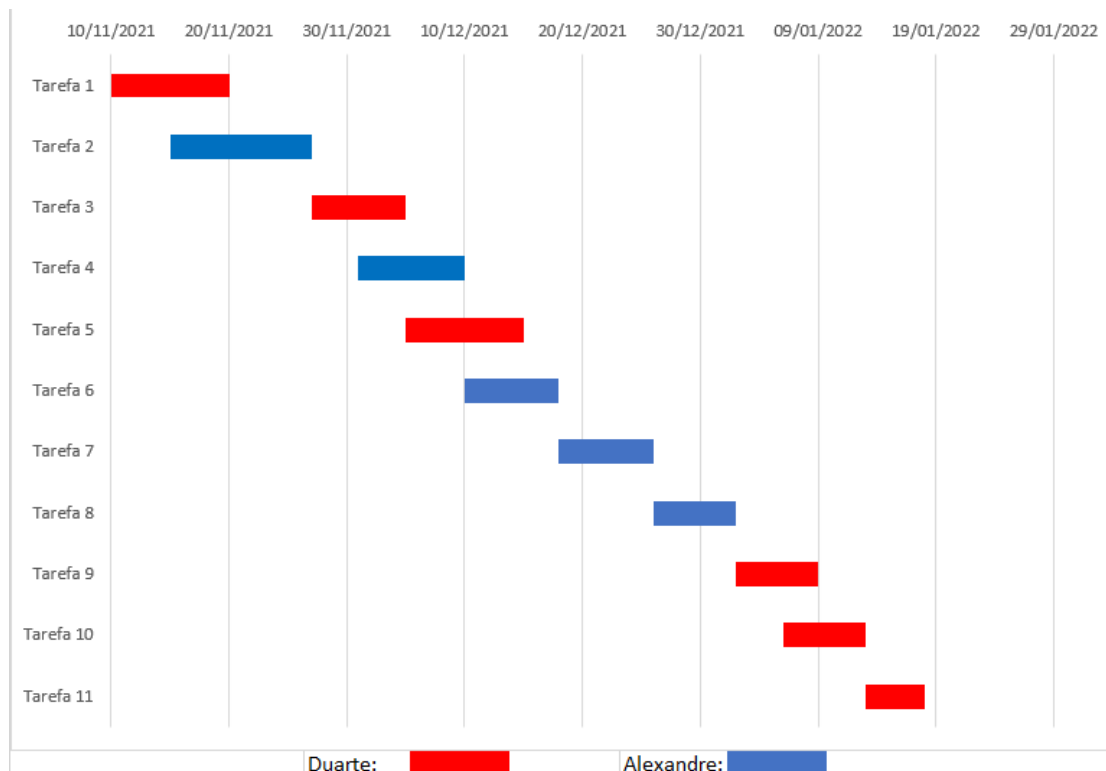
Figura 6 – Planeamento de Tarefas: PSI (Web)

Tarefas	Data início	Duração (dias)	Data de conclusão
Tarefa 1	10/11/2021	10	20/11/2021
Tarefa 2	15/11/2021	12	27/11/2021
Tarefa 3	27/11/2021	8	05/12/2021
Tarefa 4	01/12/2021	9	10/12/2021
Tarefa 5	05/12/2021	10	15/12/2021
Tarefa 6	10/12/2021	8	18/12/2021
Tarefa 7	18/12/2021	8	26/12/2021
Tarefa 8	26/12/2021	7	02/01/2022
Tarefa 9	02/01/2022	7	09/01/2022
Tarefa 10	06/01/2022	7	13/01/2022
Tarefa 11	13/01/2022	5	18/01/2022

Tarefas	Designação
Tarefa 1	Login
Tarefa 2	Registo
Tarefa 3	Rules e permissões
Tarefa 4	Layout (frontend)
Tarefa 5	Layout (backend)
Tarefa 6	CRUD's (backend)
Tarefa 7	Editar Perfil
Tarefa 8	CRUD Avaliações
Tarefa 9	Rules e permissões de Avaliações
Tarefa 10	Favoritos
Tarefa 11	Carrinho de compras

Fonte: Elaboração Própria

Figura 7 – Cronologia: PSI (Web)



Fonte: Elaboração Própria

4.4. API

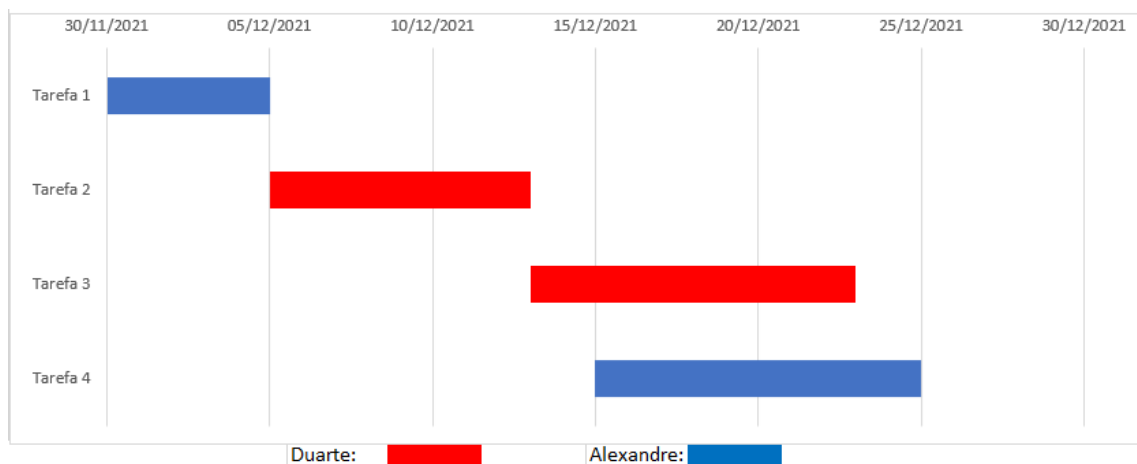
Figura 8 – Planeamento de Tarefas: SIS (API)

Tarefas	Data início	Duração (dias)	Data de conclusão
Tarefa 1	25/11/2021	10	05/12/2021
Tarefa 2	05/12/2021	8	13/12/2021
Tarefa 3	13/12/2021	10	23/12/2021
Tarefa 4	15/12/2021	10	25/12/2021

Tarefas	Designação
Tarefa 1	Mosquitto
Tarefa 2	Custom
Tarefa 3	Metodos CRUD (Controladores)
Tarefa 4	Messaging

Fonte: Elaboração Própria

Figura 9 – Cronologia: SIS (API)



Fonte: Elaboração Própria

5. Análise

5.1. 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.

5.2. 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.

5.3. REQUISITOS E FUNCIONALIDADES

5.3.1. WEBSITE (FRONTEND)

Tabela 1 - Requisitos e Funcionalidades (Frontend)

RF-01	Ver e adicionar avaliações nos produtos
-------	-----------------------------------------

RF02	Realizar <i>logout</i> , 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

5.3.2. WEBSITE (BACKEND)

Tabela 2 - Requisitos e Funcionalidades WEB(Frontend)

RF-BO-01	Realizar <i>logout</i> , 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

5.3.3. APLICAÇÃO MÓVEL

Tabela 3 - Requisitos e Funcionais Aplicação Móvel

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

5.3.4. API

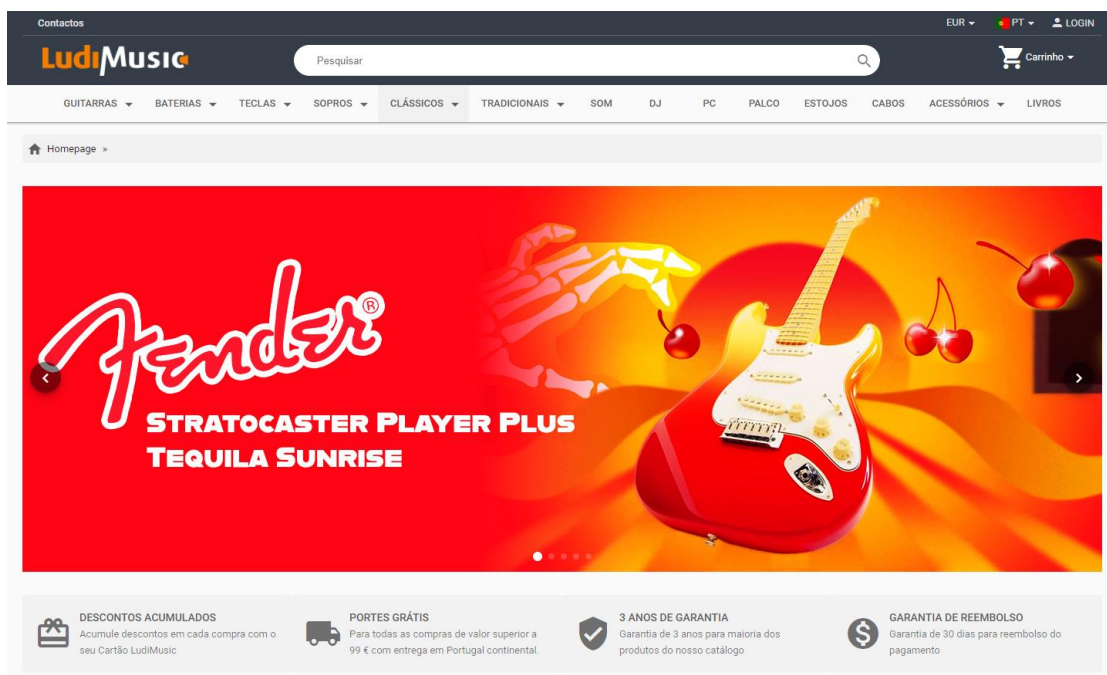
Tabela 4 - Requisitos e Funcionalidades API

RF-01	Atualização de guitarras
RF02	Gestão da autenticação

5.4. ANÁLISE CONCORRENCIAL

5.4.1. 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:	<ul style="list-style-type: none"> - Variedade para além das Guitarras - Website responsivo - Interface do website intuitiva - Sistema de pontos
Desvantagens:	<ul style="list-style-type: none"> - Demasiadas categorias - Falta de conteúdo exclusivo - Sem aplicação móvel

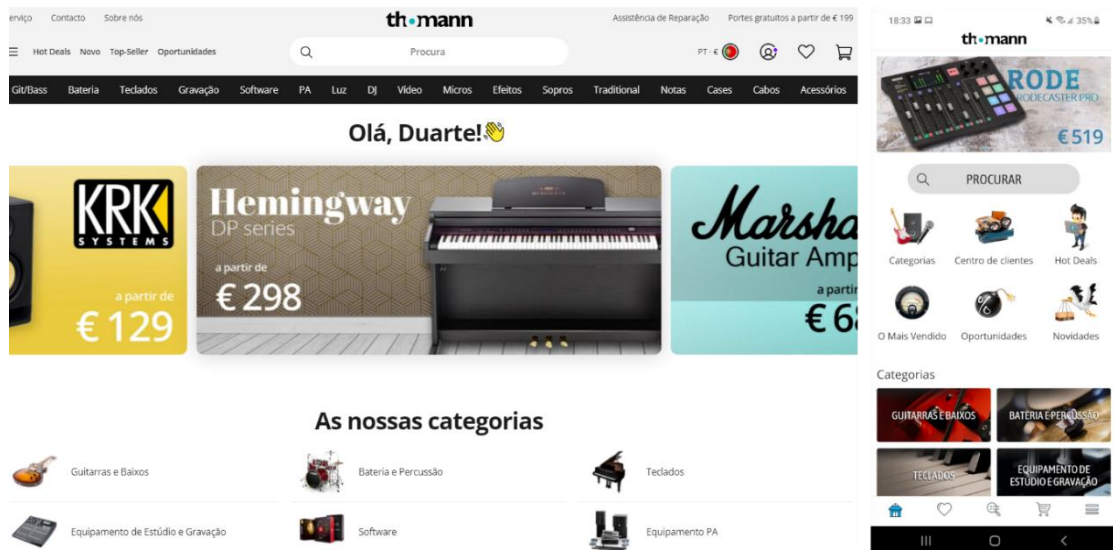
O que falta:

- Aplicação móvel
- Conteúdos exclusivos
- Mapa com lojas física

Fonte: Elaboração Própria

5.4.2. 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:	<ul style="list-style-type: none"> - 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*

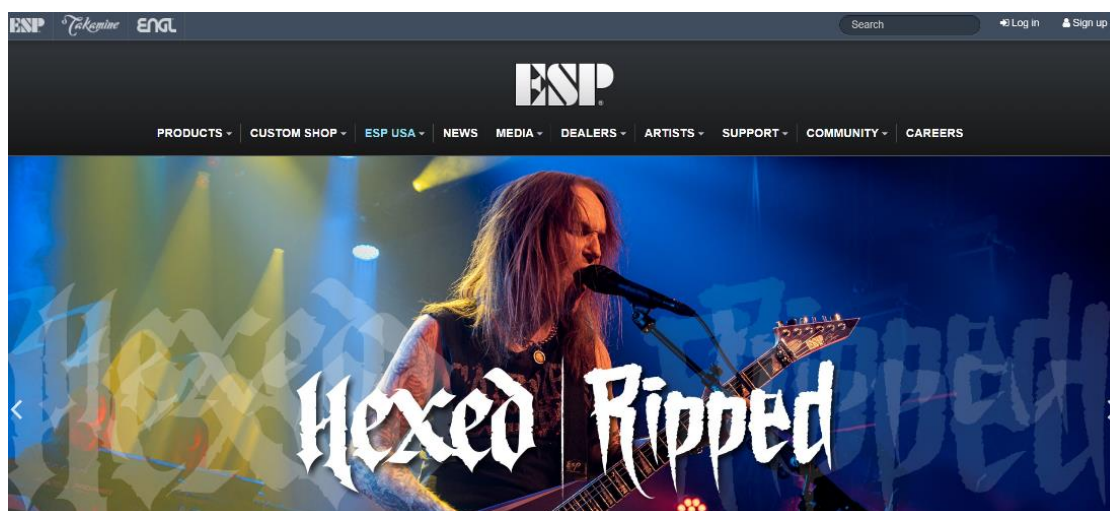
O que falta:

- 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

5.4.3. 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:	<ul style="list-style-type: none"> - <i>Website</i> responsivo - Interface do <i>website</i> intuitiva - Apenas vendem guitarras, o que simplifica o processo de compra no <i>website</i> - Galeria de fotos - Notícias
Desvantagens:	<ul style="list-style-type: none"> - Falta de conteúdo exclusivo - Sem aplicação móvel - Pouca atenção ao design
O que falta:	<ul style="list-style-type: none"> - Aplicação móvel - Sistema de pontos

Fonte: Elaboração Própria

6. 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.

6.1. **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 D](#)), ([Anexo E](#)), ([Anexo F](#)).

6.2. **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](#)).

7. Implementação

7.1. 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

7.2. 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](#))

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

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

7.3. APLICAÇÃO WEB

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

Isto traz 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.

7.3.1. 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.

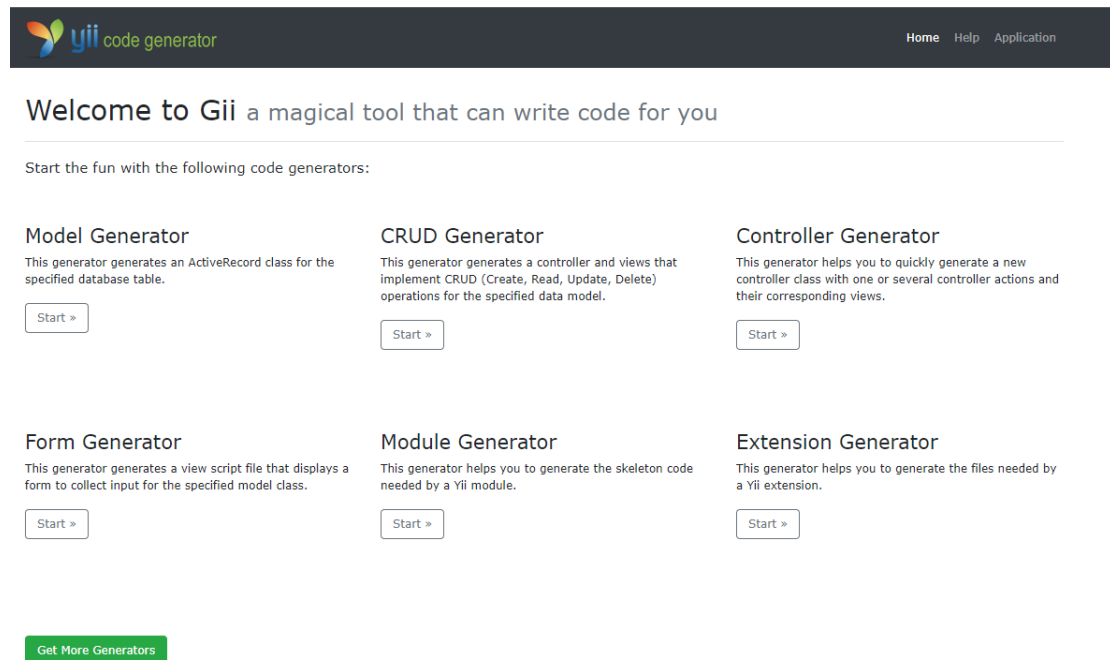
O **Controlador**, por sua vez, efetua a gestão dos dois mencionados anteriormente, ao receber os dados vindos do Modelo, processando-os, e, de seguida, enviando-os para a Vista.

7.3.2. 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

7.3.3. 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 definidas 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”

```
$crudUsers = $auth->createPermission( name: 'crudUsers');
$crudUsers->description = 'CRUD Users';
$auth->add($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

```
//admin
$admin = $auth->createRole( name: 'admin');
$auth->add($admin);
$auth->addChild($admin, $funcionario);
$auth->addChild($admin, $gestor);
$auth->addChild($admin, $cliente);
$auth->addChild($admin, $crudUsers);
```

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

```
if (\Yii::$app->user->can( permissionName: 'crudtabelaGuitarras')) {  
    $menuItems[] =['label' => 'Guitarras', 'icon' => 'fas fa-guitar', 'url' => ['guitarras/index']];  
}
```

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

```
'access' => [  
    'class' => AccessControl::className(),  
    'rules' => [  
        [  
            'actions' => ['index','update', 'delete', 'create', 'view'],  
            'allow' => true,  
            'roles' => ['admin'],  
        ],  
    ],  
],
```

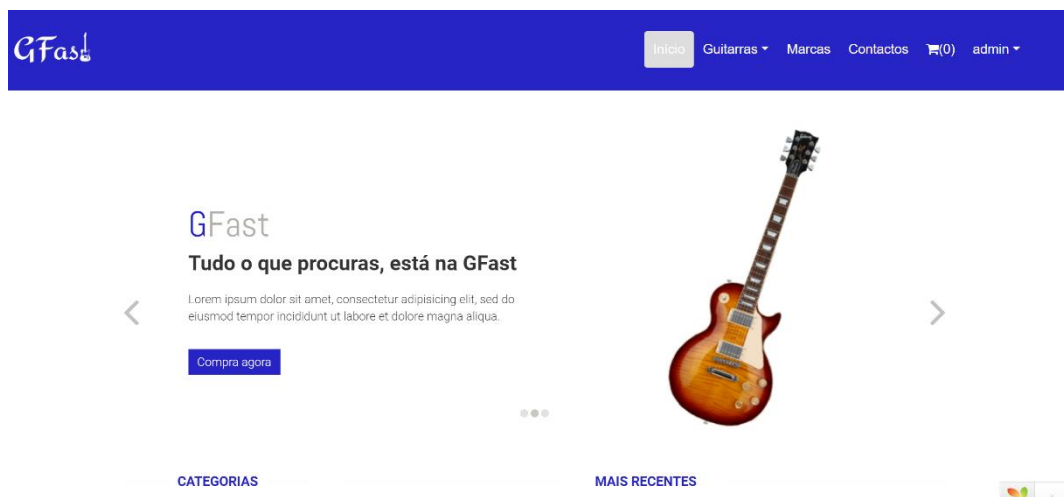
Fonte: Elaboração Própria

7.3.4. 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”.

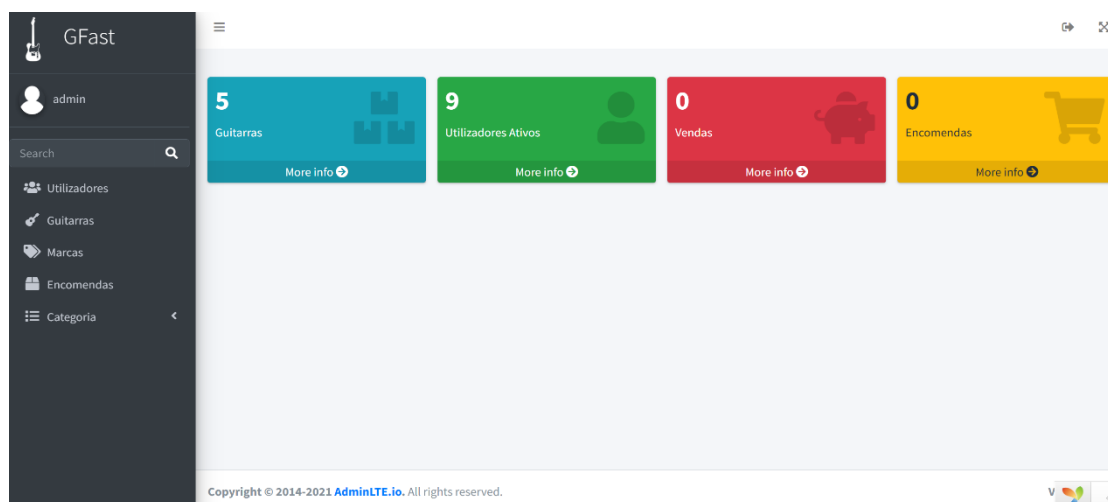
Figura 19 - Template Frontend



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

7.3.5. EXTRAS

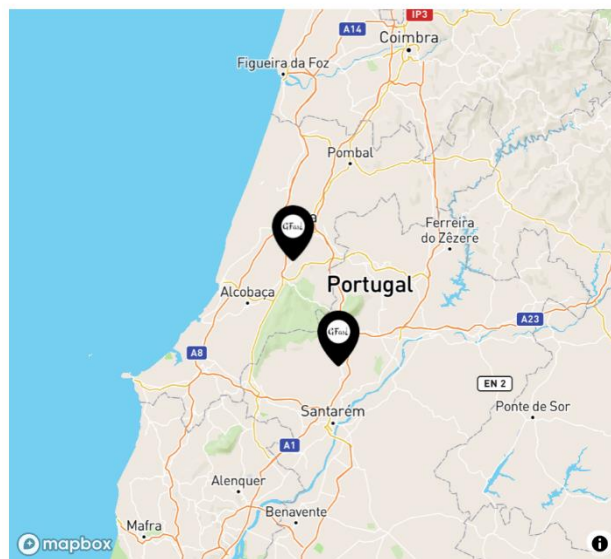
Na página de contato foi criado um mapa como extra com as diversas localizações fictícias das lojas físicas da “GFast”. Para a criação deste mapa é chamada uma API, denominada mapbox, que cria o mapa e, através de coordenadas GPS, cria pontos no mapa.

Figura 21 - Criação de pontos no mapa através de Coordenadas

```
var geojson =
{
  "type": "FeatureCollection",
  "features": [{
    "type": "Feature",
    "geometry": {"type": "Point", "coordinates": [-8.82098965175199, 39.73457126808567]},
    "properties": "Sede GFast"
  },
  {
    "type": "Feature",
    "geometry": {"type": "Point", "coordinates": [-8.666880016033495, 39.459549691983646]},
    "properties": "GFast Alcanena"
  }
  ]
};
```

Fonte: Elaboração Própria

Figura 22 - Mapa da localização de lojas



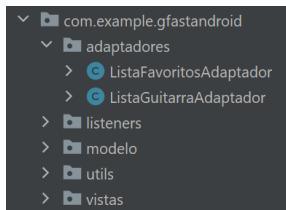
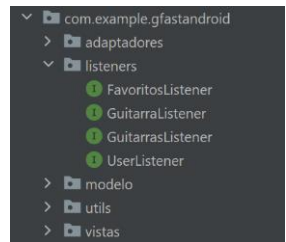
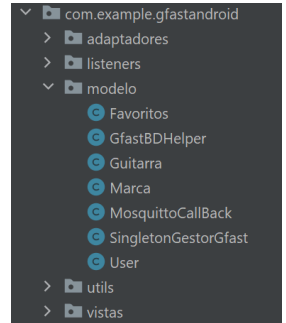
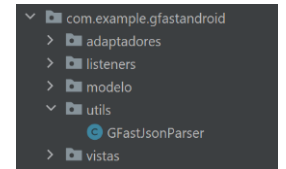
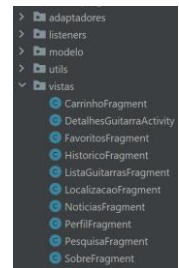
Fonte: Elaboração Própria

7.4. 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 <i>callback</i> 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

7.4.1. 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

```
//-----GUITARRAS-----
public void getAllGuitarrasAPI(final Context context) {

    if (!GfastJsonParser.isConnectedInternet(context)) {
        Toast.makeText(context, "NÃO tem ligação à rede", Toast.LENGTH_SHORT).show();
    }

    if (guitarrasListener != null) {
        guitarrasListener.onRefreshListaGuitarras(gfastBDHelper.getAllGuitarrasBD());
    }
} else {
    //GET todas as guitarras da aplicação
    JSONArrayRequest request = new JSONArrayRequest(Request.Method.GET, urlAPIGfast, jsonRequest: null, new Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONArray response) {
            try {
                //Converter JSON vindo da API para um objeto
                guitarras = GfastJsonParser.parserJsonGuitarras(response);
                adicionarGuitarrasBD(guitarras);

                if (guitarrasListener != null) {
                    //Atualizar a lista de guitarras
                    guitarrasListener.onRefreshListaGuitarras(guitarras);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            //Caso o pedido à API dê erro
            Toast.makeText(context, error.getMessage(), Toast.LENGTH_SHORT).show();
            System.out.println(error.getMessage());
        }
    });

    volleyQueue.add(request);
}
```

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

```
//Ir buscar todas as guitarras à bd
public ArrayList<Guitarra> getGuitarras() {
    guitarras = gfastBDHelper.getAllGuitarrasBD();
    return guitarras;
}

//Buscar uma guitarra à DB
public Guitarra getGuitarraBD(int id) {

    for (Guitarra g : guitarras) {

        if (g.getGui_id() == id) {
            return g;
        }
    }

    return null;
}
```

Fonte: Elaboração Própria

7.4.2. 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.

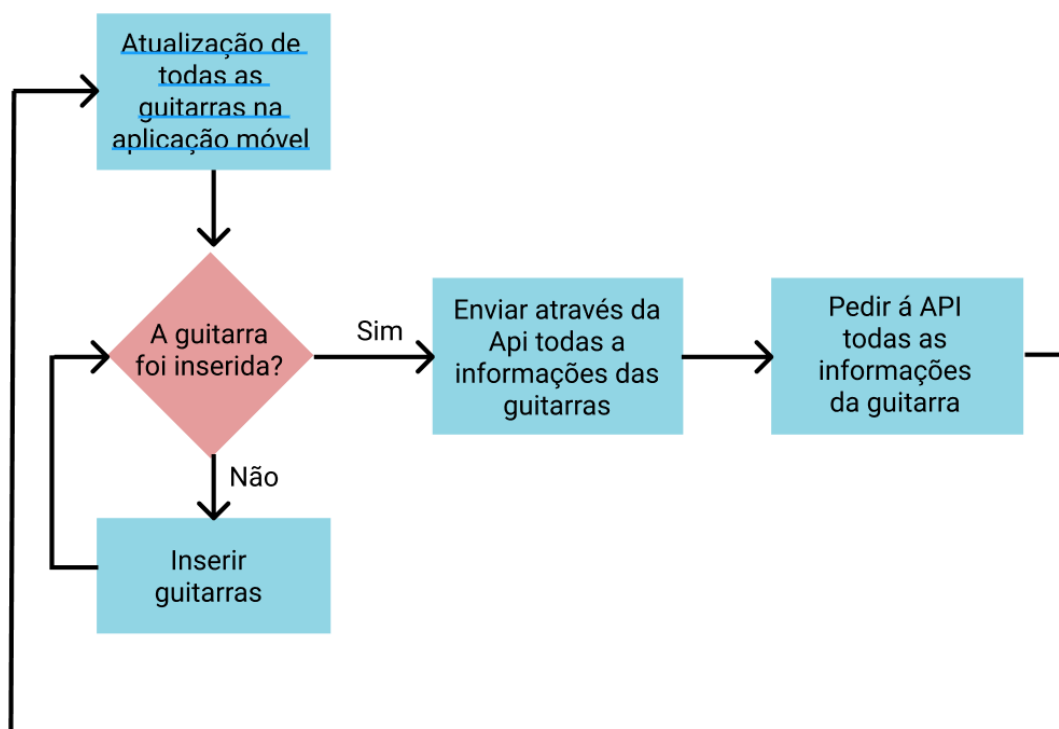
- Nesta lista é possível ao pressionar uma guitarra abrir os seus detalhes.

7.5. 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.

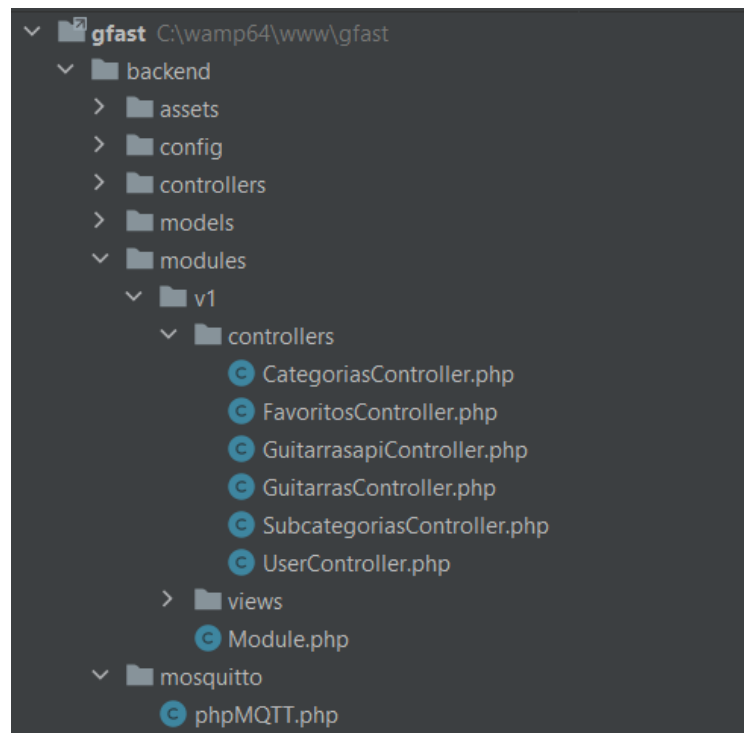
Figura 25 – implementação da API



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

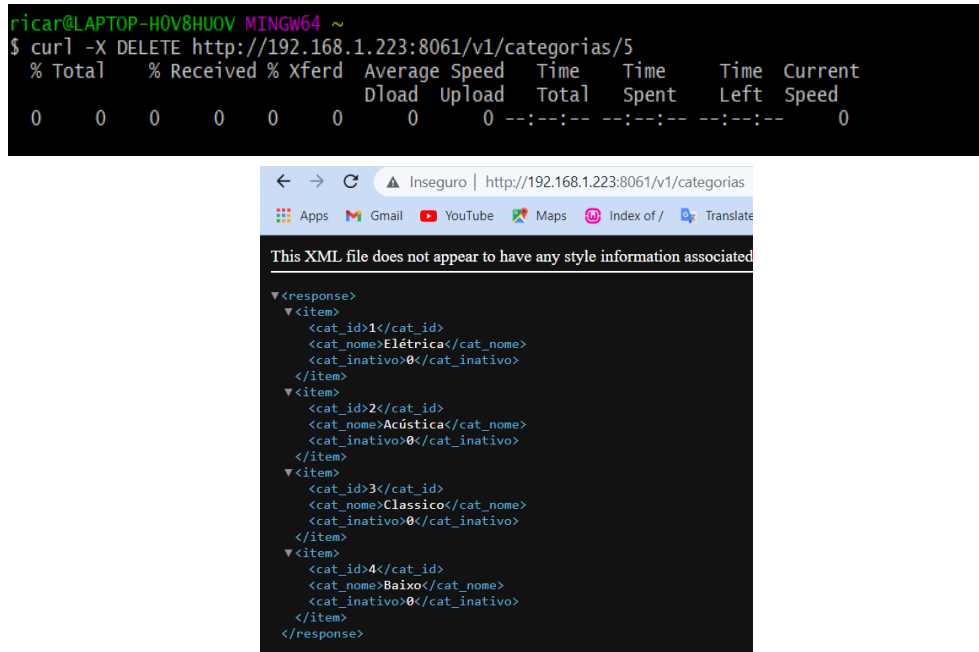
7.5.1. cURL resultado no CMD

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

7.5.1.1. Categorias

GET – Categorias

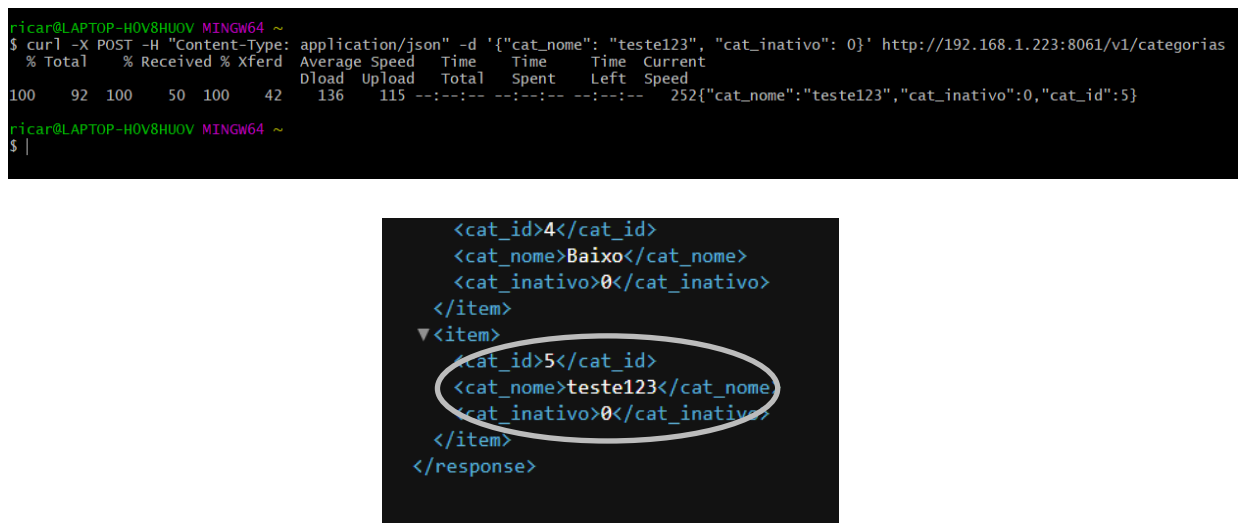
Figura 27 – Exemplo GET Categorias



Fonte: Elaboração Própria

POST – Categorias

Figura 28 – Exemplo POST Categorias



Fonte: Elaboração Própria

PUT – Categorias

Figura 29 – Exemplo PUT Categorias

```
ricar@LAPTOP-HOV8HUOV MINGW64 ~  
$ curl -X PUT -H "Content-Type: application/json" -d '{"cat_nome": "teste", "cat_inativo": 0}' http://192.168.1.223:8061/v1/categorias/5  
% Total % Received % Xferd Average Speed Time Time Time Current  
Dload Upload Total Spent Left Speed  
100 86 100 47 100 39 164 136 --:--:-- --:--:-- --:--:-- 301["cat_id":5,"cat_nome":"teste","cat_inativo":0]  
ricar@LAPTOP-HOV8HUOV MINGW64 ~  
$ |
```

```
</item>  
▼<item>  
  <cat_id>4</cat_id>  
  <cat_nome>Baixo</cat_nome>  
  <cat_inativo>0</cat_inativo>  
</item>  
▼<item>  
  <cat_id>5</cat_id>  
  <cat_nome>teste</cat_nome>  
  <cat_inativo>0</cat_inativo>  
</item>  
</response>
```

Fonte: Elaboração Própria

DELETE – Categorias

Figura 30 – Exemplo DELETE Categorias

```
ricar@LAPTOP-HOV8HUOV MINGW64 ~  
$ curl -X DELETE http://192.168.1.223:8061/v1/categorias/5  
% Total % Received % Xferd Average Speed Time Time Time Current  
Dload Upload Total Spent Left Speed  
0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0
```

```
<?xml version="1.0" encoding="UTF-8" ?>  
<response>  
  <name>Not Found</name>  
  <message>Object not found: 5</message>  
  <code>0</code>  
  <status>404</status>  
  <type>yii\web\NotFoundHttpException</type>  
</response>
```

Fonte: Elaboração Própria

7.5.2. 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

1. Users

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

2. Guitarras

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

3. Subcategorias

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

3.1.1. Categorias

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

7.5.3. 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

```
public function afterSave($insert, $changedAttributes)
{
    parent::afterSave($insert, $changedAttributes);
    //Obter dados do registro em causa
    $id = $this->gui_id;
    $marca = $this->gui_idmarca;
    $subcategoria = $this->gui_idsubcategoria;
    $idreferencia = $this->gui_idreferencia;
    $descricao = $this->gui_descricao;
    $nome = $this->gui_nome;
    $preco = $this->gui_preco;
    $iva = $this->gui_iva;
    $inativo = $this->gui_inativo;
    $fotopath = $this->gui_fotopath;
    $qrcodepath = $this->gui_qrcodepath;

    $myObj=new \stdClass();
    $myObj->id=$id;
    $myObj->subcategoria=$subcategoria;
    $myObj->marca=$marca;
    $myObj->iva=$iva;
    $myObj->preco=$preco;
    $myObj->nome=$nome;
    $myObj->idreferencia=$idreferencia;
    $myObj->descricao=$descricao;
    $myObj->fotopath=$fotopath;
    $myObj->qrcodepath=$qrcodepath;
    $myObj->inativo=$inativo;
}
```

```
$myJSON = json_encode($myObj);
if($insert)
    $this->FazPublishNoMosquitto( canal: "INSERT",$myJSON);
else
    $this->FazPublishNoMosquitto( canal: "UPDATE",$myJSON);
}

public function afterDelete()
{
    parent::afterDelete();
    $prod_id= $this->gui_id;
    $myObj=new \stdClass();
    $myObj->id=$prod_id;
    $myJSON = json_encode($myObj);
    $this->FazPublishNoMosquitto( canal: "DELETE",$myJSON);
}

public function FazPublishNoMosquitto($canal,$msg)
{
    $server = "broker.emqx.io";
    $port = 1883;
    $username = ""; // set your username
    $password = ""; // set your password
    $client_id = "PHP"; // unique!
    $mqtt = new phpMQTT($server, $port, $client_id);
    if ($mqtt->connect( clean: true, will: NULL, $username, $password))
    {
        $mqtt->publish($canal, $msg, qos: 1);
        $mqtt->close();
    }
    else { file_put_contents( filename: 'debug.output', data: "Time out!"); }
}
```

Fonte: Elaboração Própria

8. Testes

8.1. 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

```
Backend\tests.unit Tests (9) -----
+ CategoriasTest: Validar categoria (0.06s)
+ CategoriasTest: Criar categoria (0.03s)
+ CategoriasTest: Editar categoria (0.04s)
+ GuitarrasTest: Validar guitarra (0.11s)
+ GuitarrasTest: Criar guitarra (0.72s)
+ GuitarrasTest: Editar guitarra (0.73s)
+ MarcasTest: Validar marca (0.03s)
+ MarcasTest: Criar marca (0.04s)
+ MarcasTest: Editar marca (0.04s)
```

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

```
Backend\tests.functional Tests (11) -----
+ CategoriasCrudCest: Test inserir categoria (0.74s)
+ CategoriasCrudCest: Test editar categoria (0.68s)
+ GuitarraCrudCest: Test inserir guitarra (0.00s)
+ GuitarraCrudCest: Test editar guitarra (0.00s)
+ LoginCest: Login user (0.68s)
+ MarcaCrudCest: Test inserir marca (0.71s)
+ MarcaCrudCest: Test editar marca (0.68s)
+ SubcategoriaCrudCest: Test inserir subcategoria (0.68s)
+ SubcategoriaCrudCest: Test editar subcategoria (0.69s)
+ UserCrudCest: Create user (1.37s)
8.2 + UserCrudCest: Edit user (0.72s)
-----
```

Fonte: Elaboração Própria

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

```
Frontend\tests.unit Tests (7) -----
+ AvaliacoesTest: Avaliacao (0.27s)
+ AvaliacoesTest: Criar registo (0.11s)
+ AvaliacoesTest: Editar user (0.10s)
+ RegistoTest: Registo (0.08s)
+ RegistoTest: Registo correto (0.67s)
+ RegistoTest: Editar user (0.04s)
+ RegistoTest: Email incorreto (0.11s)
-----
```

Fonte: Elaboração Própria

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

```
Frontend\tests.functional Tests (7) -----
+ LoginCest: Check empty (0.08s)
+ LoginCest: Check wrong password (0.03s)
+ LoginCest: Check inactive account (0.02s)
+ LoginCest: Check valid login (0.01s)
+ SignupCest: Registo campos vazios (0.05s)
+ SignupCest: Registo campos incorretos (0.05s)
+ SignupCest: Registo correto (0.02s)
-----
```

Fonte: Elaboração Própria

Testes de Aceitação e objetivos

Tabela 14 – Frontend: Testes de Aceitação

	AvaliacaoCest
--	----------------------

1	<p>Nome: CRUDAvaliacao</p> <p>Objetivo: Teste para inserir, editar e apagar uma avaliação de um utilizador.</p>
---	-----------------------------------------------------------------------------------------------------------------

Fonte: Elaboração Própria

Figura 36 – Teste complete de aceitação

```
{
  $I->amOnPage( page: '/');
  $I->click( link: 'Entrar');
  $I->seeCurrentURLEquals( uri: '/site/login');
  $I->fillField( field: 'LoginForm[username]', value: 'admin');
  $I->fillField( field: 'LoginForm[password]', value: 'admin123');
  $I->click( link: 'login-button');
  $I->seeCurrentURLEquals( uri: '/');
  $I->amOnPage( page: '/site/produto?id=5');
  $I->seeCurrentURLEquals( uri: '/site/produto?id=5');

  $I->click( link: '/html/body/main/div/main/section/a');
  $I->seeCurrentURLEquals( uri: '/avaliacoes/create?id=5');
  $I->fillField( field: 'Avaliacoes[ava_avaliacao]', value: 'Era Mesmo esta que eu queria!!');
  $I->click( link: 'Save');
  $I->click( link: '/html/body/main/div/main/section/div[2]/div[1]/a[1]/i');

  $I->fillField( field: 'Avaliacoes[ava_avaliacao]', value: 'É Mesmo esta que eu quero!!');
  $I->click( link: 'Save');

  $I->see( text: "É Mesmo esta que eu quero!!");

  $I->click( link: "/html/body/main/div/main/section/div[2]/div/a[2]");
  $I->acceptPopup();

  $I->dontSee( text: "É Mesmo esta que eu quero!!");
}
```

Fonte: Elaboração Própria

9. 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 foram 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.

10. 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_-S050cPV7goWgqq3O6ABU0JqHHB5vmYakz4aAtsaEALw_wcB
- Thomann. (2022). *Thomann*. <https://www.thomann.de/gb/index.html>

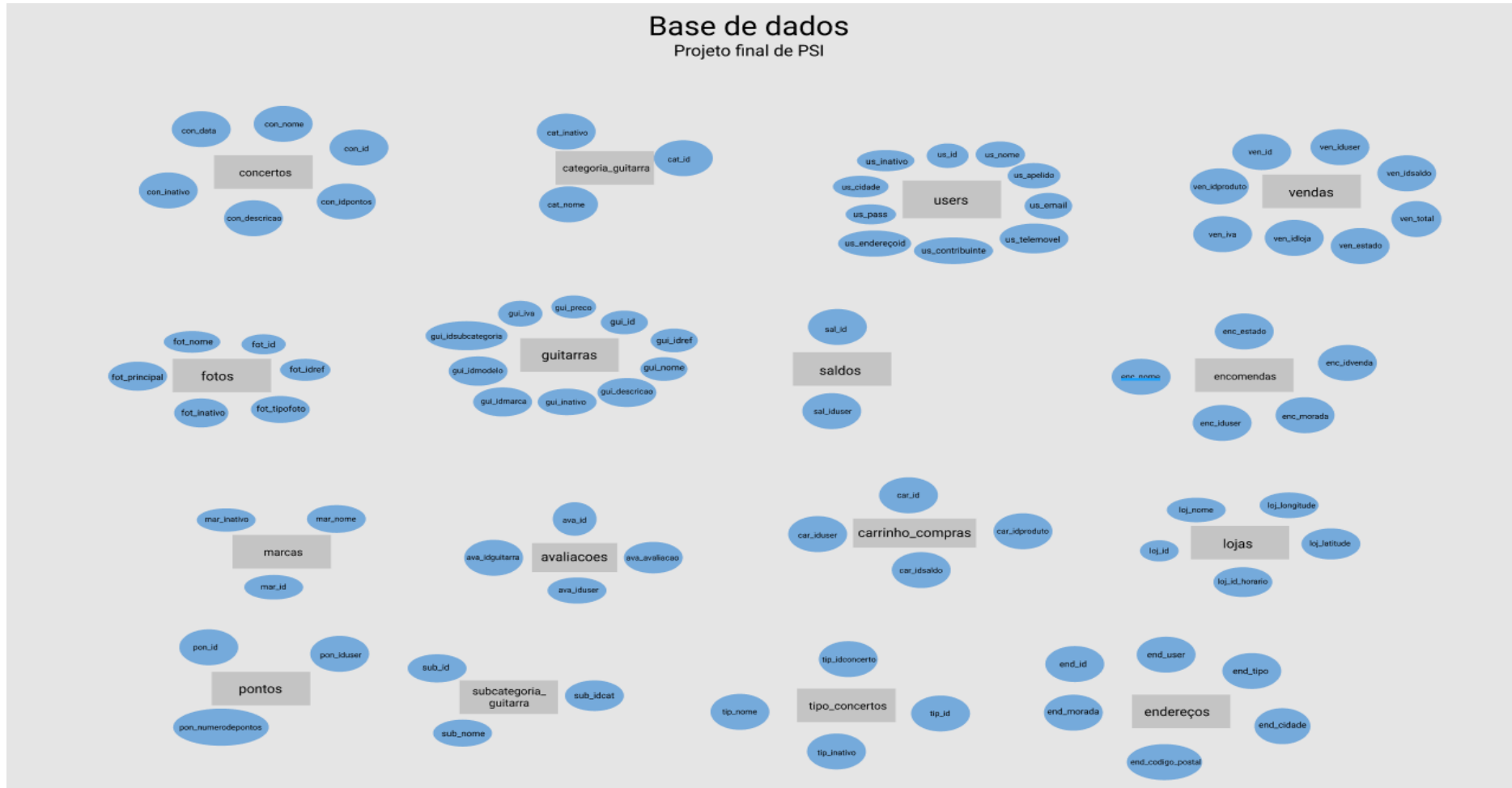
Anexos

Acessos ao projeto

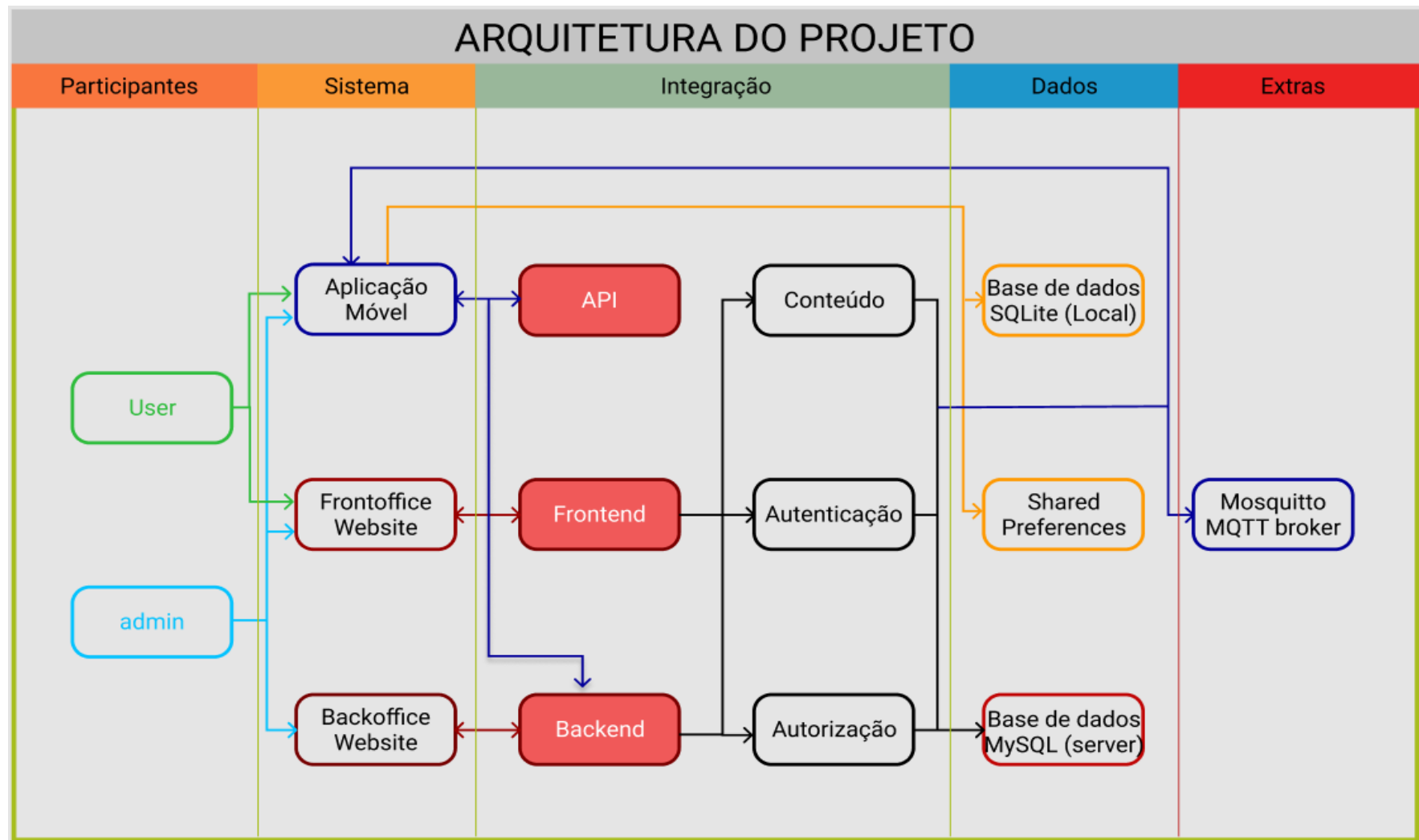
Username	Pasword
cliente	1234567890
func	1234567890
gestor	1234567890
admin	admin123

Anexo A

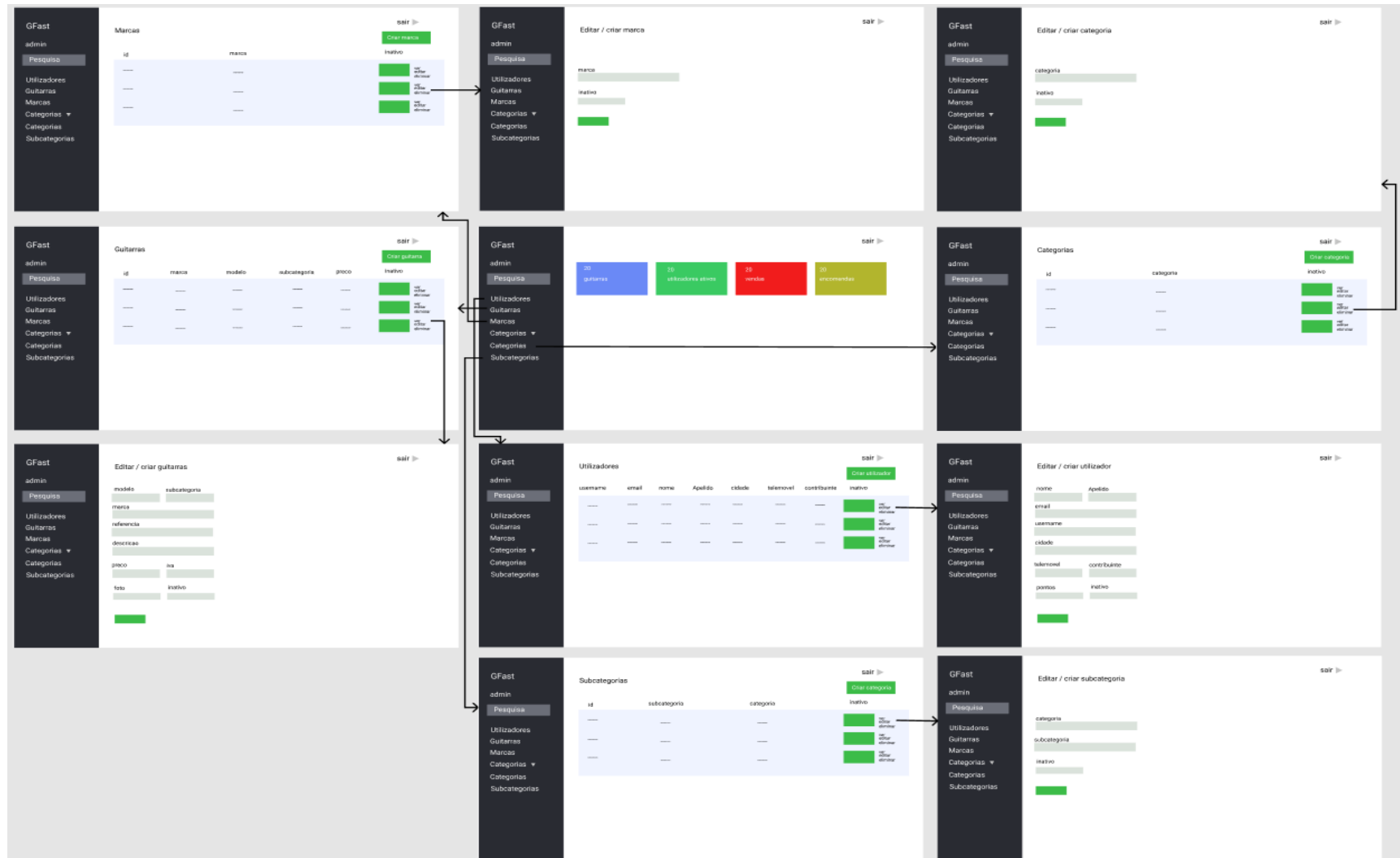
Base de dados Projeto final de PSI



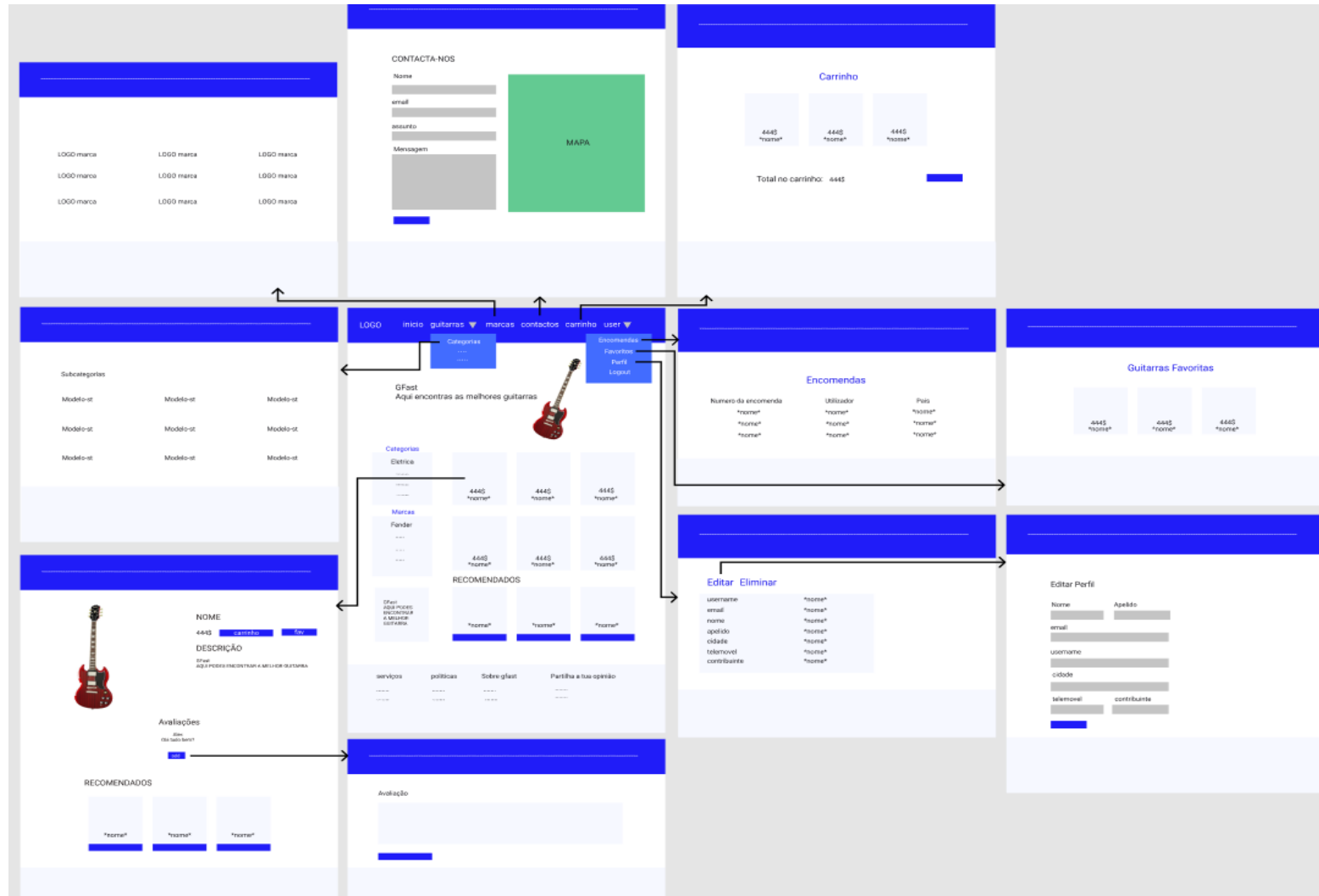
Anexo B



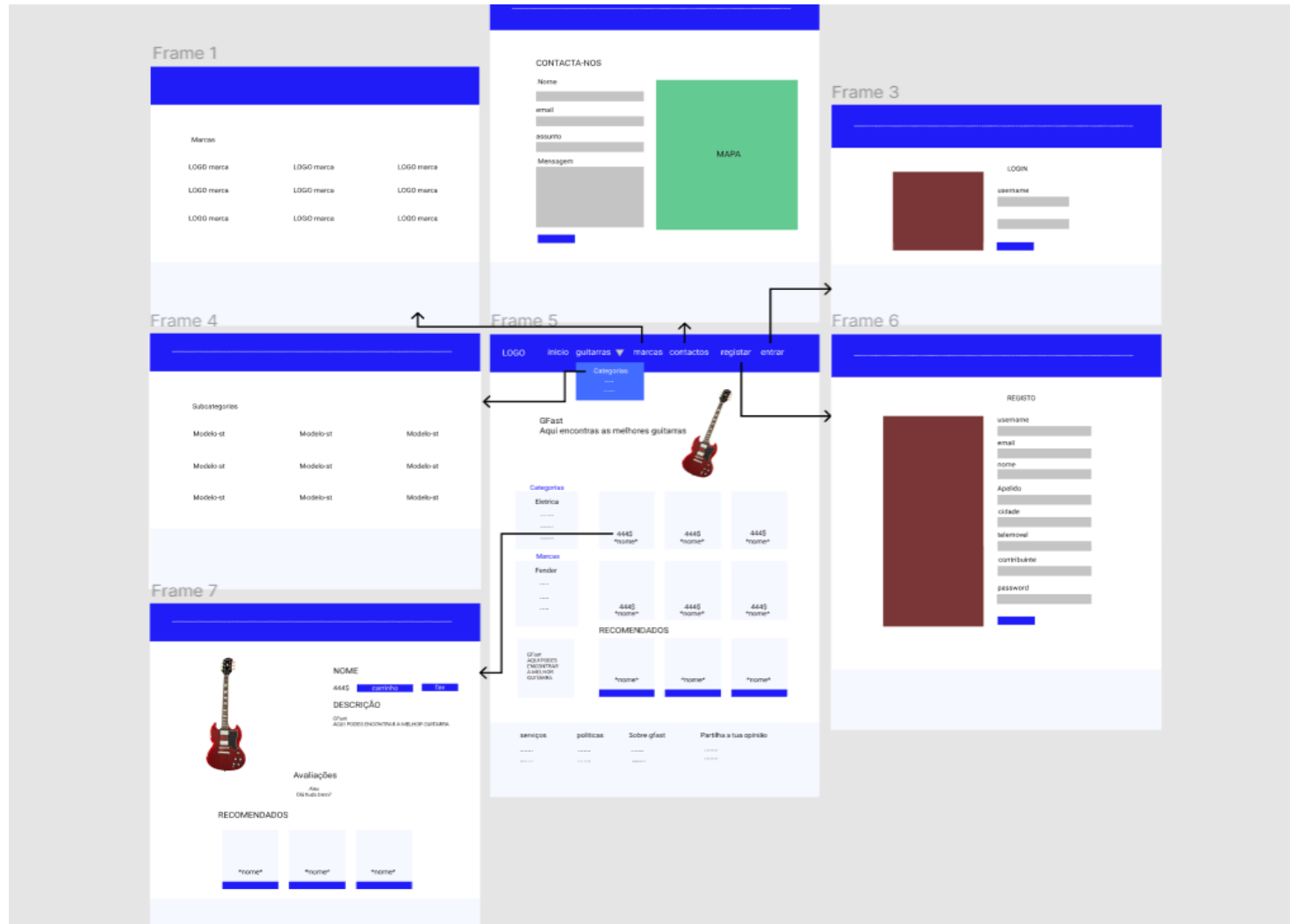
Anexo C



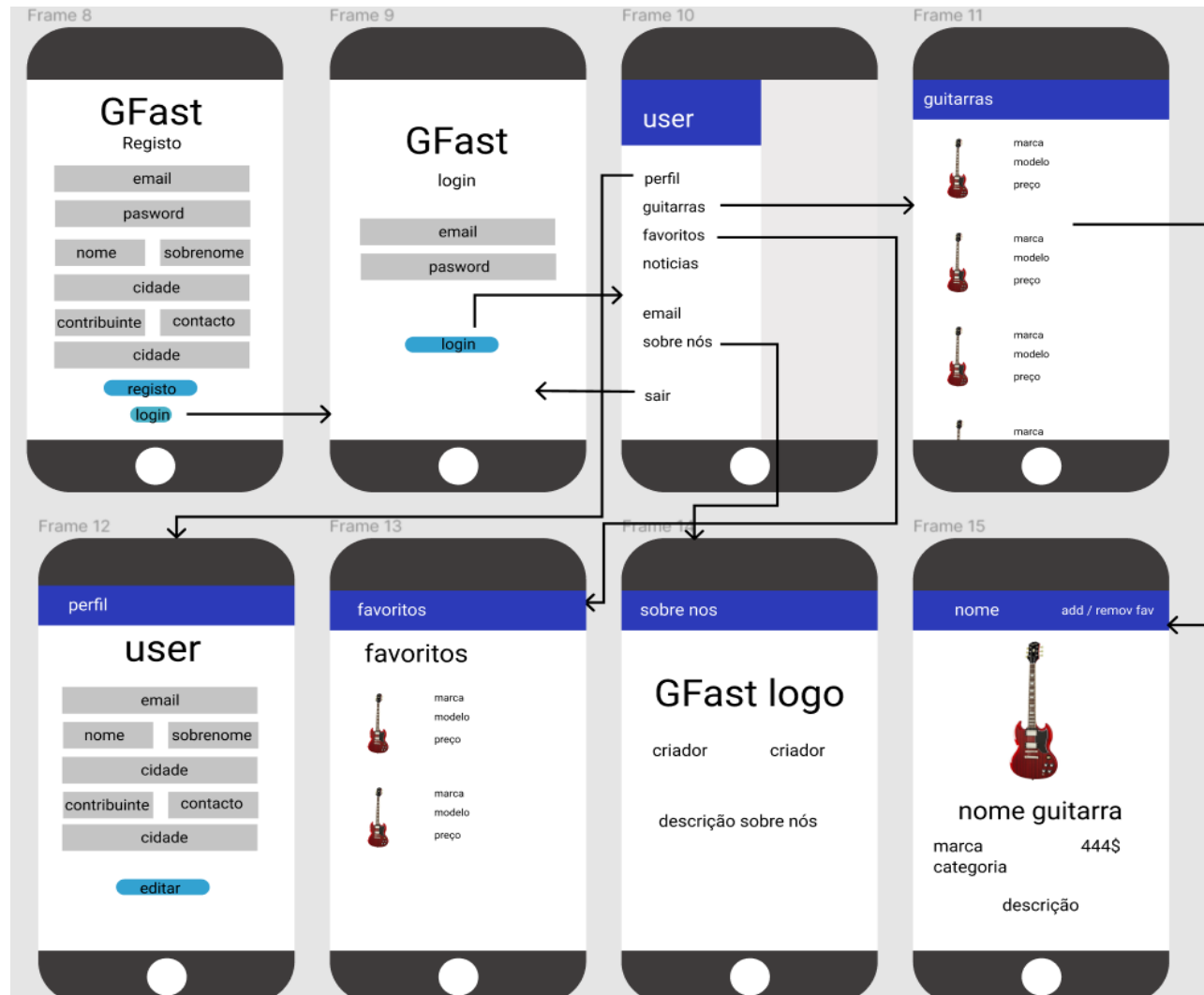
Anexo D



Anexo E



Anexo F



Anexo G

