 **INSTITUTO POLITÉCNICO DE BEJA **

**Escola Superior de Tecnologia e Gestão**

**Tecnologias Web e Dispositivos Móveis**

**Programação de Aplicação para Servidor**

**Aplicação Android para a compras de jogos online**

**Elaborado por:**

**Dylan Fialho**

**Tiago Miguel Nunes Caracol**

Beja

17/06/2021

 **INSTITUTO POLITÉCNICO DE BEJA **

**Escola Superior de Tecnologia e Gestão**

**Tecnologias Web e Dispositivos Móveis**

**Programação de Aplicação para Servidor**

**Aplicação Android para a compras de jogos online**

**Elaborado por:**

**Dylan Fialho**

**Tiago Miguel Nunes Caracol**

Docente:

Luís Rosário e Carlos Sancho

Beja

17/06/202

Resumo

Este relatório, elaborado no âmbito do Curso de Tecnologias Web e Dispositivos Móveis expõe as nossas competências adquiridas ao longo deste semestre nas disciplinas de Programação para dispositivos móveis 2 e Programação para web.

A escolha deste projeto:

Aplicação Android para a compras de jogos online-> utilizamos o mesmo tema de trabalho relativo ao semestre passado do aluno Dylan Fialho, chegando ao acordo mútuo sobre o tema do trabalho.

Índice Geral

[1 Introdução 1](#_Toc95833250)

[2 Organização do trabalho e ferramentas 2](#_Toc95833251)

[3 Análise do problema 3](#_Toc95833252)

[4 Funcionalidades da aplicação 4](#_Toc95833253)

[6 Base de dados 5](#_Toc95833254)

[5 Interface da aplicação 6](#_Toc95833255)

[5 Interface da aplicação 7](#_Toc95833256)

[7 Partes importantes do código 8](#_Toc95833257)

[7 Partes importantes do código 9](#_Toc95833258)

[7 Partes importantes do código 11](#_Toc95833259)

[8 Conclusão 13](#_Toc95833260)

[9 Referências Bibliográficas 14](#_Toc95833261)

# Introdução

O presente projeto pertence à disciplina de PAS (Programação de Aplicação para servidor do Curso de Tecnologias Web e Dispositivos Móveis do IPBeja. Este tem como principal objetivo abranger todos os conhecimentos adquiridos ao longo deste semestre.

O tema escolhido foi Aplicação Android para a compras de jogos online e este tem como objetivo o desenvolvimento de uma aplicação para a compra de jogos online de diferentes plataformas.

A aplicação deverá ser capaz de dar a oportunidade ao utilizador de poder efetuar o registo e o respetivo login e eventualmente comprar os jogos do momento para qualquer plataforma.

A aplicação contém um bottom menu com várias funcionalidades, desde a página principal, um menu de categorias e o carrinho. Na página principal o utilizar pode pesquisar o jogo que pretende através de vários campos como o nome do jogo, a categoria a que ele pertence ou mesmo filtrar por plataforma.

Uma imagem com texto

Descrição gerada automaticamente

Figura 1- Smartphone

# Organização do trabalho e ferramentas

Com o tema do trabalho definido, estava na altura de começar em grupo a pensar nos layouts que iriamos utilizar na aplicação.

Em conjunto começamos por fazer pequenos esboços em papel, até futuramente replicarmos no Abode XD. Quando concluímos essa etapa, começamos por replicar os diferentes layouts no android Studio usando a matéria lecionada nas aulas de pdm2.

A seguir, foi a altura de criar a Api em laravel lecionada na disciplina de programação de aplicação ao servidor e os respetivos objetos que usamos no trabalho, concluindo isso, dividimos o resto do trabalho em objetivos e cada um de nós ficou encarregue de fazer a sua parte consoante o que se sentia mais confortável. A partir daí o trabalho foi mútuo ajudando-nos um ao ouro nas dúvidas que tínhamos.

Com o projeto quase finalizados, reunimo-nos novamente e em conjunto resolvemos os últimos problemas que achava mos que deviam ser resolvidos.

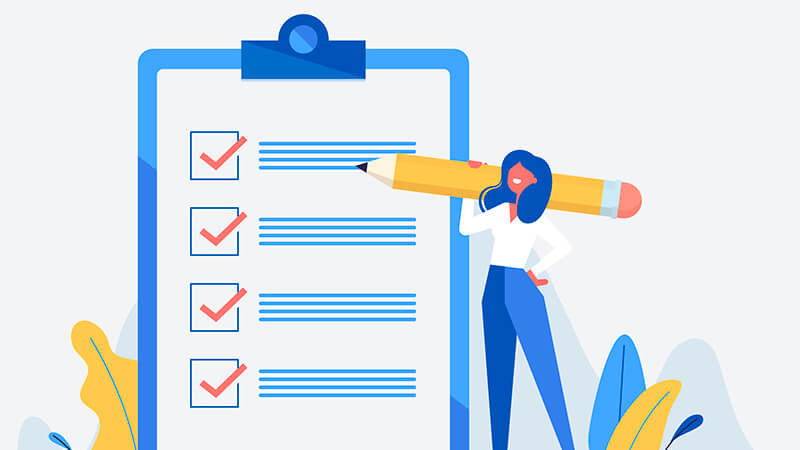


Figura 2- Tarefas

**As ferramentas utilizadas na produção deste trabalho foram:**

Android Studio- Aplicação para desenvolver código e os layouts.

Adobe Ilustrator- Aplicação onde usamos para editar os nossos ícones do menu.

GitHub- Aplicação para descarregar as novas versões.

Laravel- Aplicação que usamos para criar a nossa api.

Xamp- Aplicação que faz o host local.

# Análise do problema

Com as diversas lojas de jogos online, fizemos uma pesquisa para tentar construir uma nova aplicação que eliminava todos os problemas que nós não gostamos nas outras aplicações, reunindo tudo o que nos fazia mais sentido para poder criar uma aplicação que satisfaça todas as necessidades dos utilizadores e que seja fácil e intuitiva de navegar.

Uma imagem com texto

Descrição gerada automaticamente

Figura 3- Homem a pensar

# Funcionalidades da aplicação

* Registar um utilizador novo
* Fazer o login dentro da aplicação
* Consultar todos os jogos;
* Pesquisa de jogos através do nome do jogo;
* Pesquisa de jogos através de categorias;
* Pesquisa de jogos através das plataformas;
* Adicionar jogos ao carrinho;
* Remover jogos do carrinho;
* Comprar jogos;
* Ler as descrição de cada jogo

# Base de dados

**Base de dados no laravel (Back office)**

Uma imagem com texto

Descrição gerada automaticamente

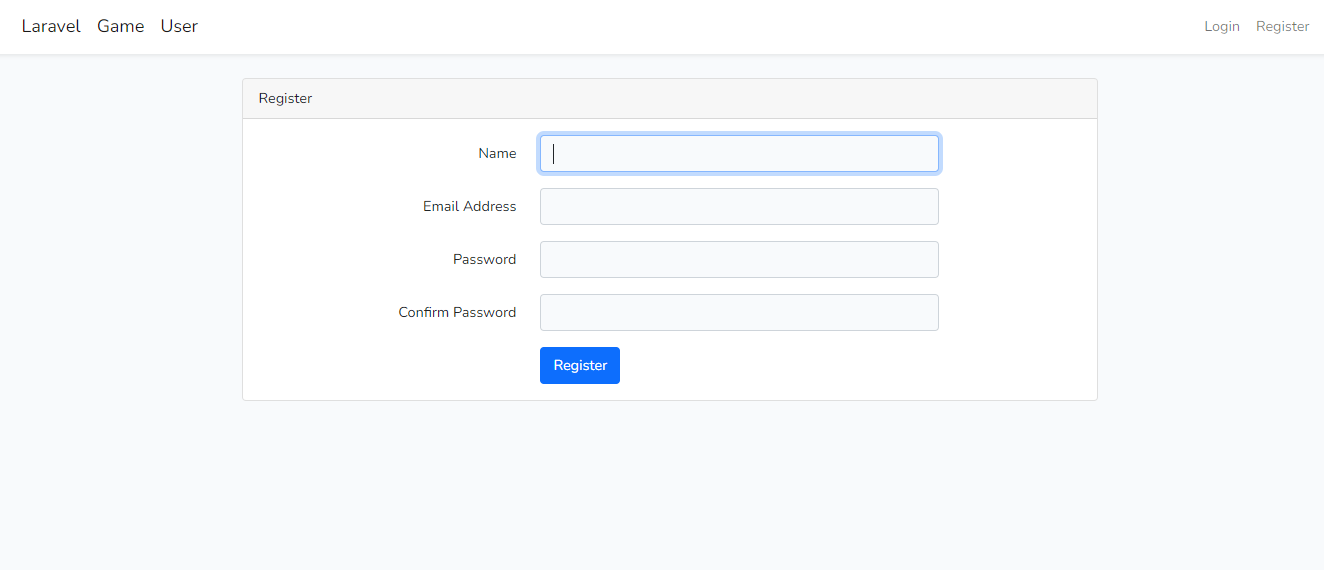
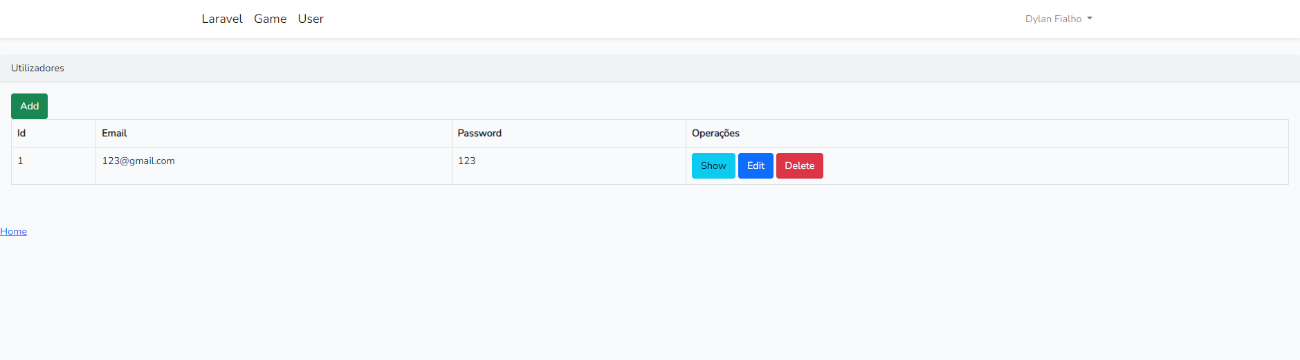


Figura - Login e registo do back office



Uma imagem com texto

Descrição gerada automaticamente

Figura - Tabela de utilizadores e tabela dos jogos

# Interface da aplicação

**Login, Registar:**

* Nesta página o utilizador pode procurar efetuar o login, ou então, caso ainda não tenha conta, o utilizador pode clicar na TextView da Activity e será readicionado para uma Activity para Registar

Uma imagem com texto

Descrição gerada automaticamente Uma imagem com texto

Descrição gerada automaticamente

Figura - Activity do login e do registar

# Interface da aplicação

**Publicidade:**

* Depois do utilizador fazer o seu login, antes de ir para a página principal da aplicação, será mostrado uma pequena publicidade sobre as regalias da nossa aplicação.

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente

Figura -Exemplo das publicidades

**Página principal :**

* A página principal apresenta uma barra de pesquisa onde o utilizador poderá procurar um jogo por diferentes campos;
* Dá a oportunidade de o utilizador poder filtrar os jogos que procurar pelas diferentes plataformas apresentadas na activity

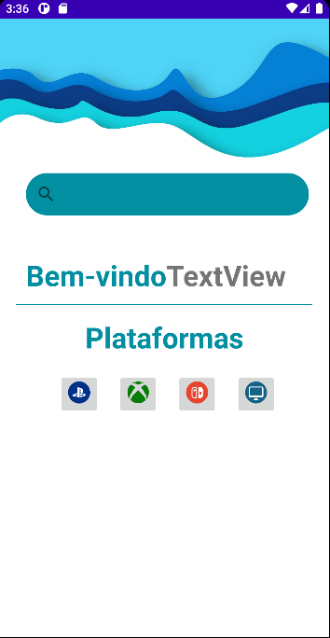


Figura - Página principal da app

# Partes importantes do código

public static User getActiveSession(Context *context*) {  
 if (**getInstance**(*context*).contains(KEY\_EMAIL) && **getInstance**(*context*).contains(KEY\_PASSWORD)) {  
 return new User(**getInstance**(*context*).getLong(KEY\_ID, 0),  
 **getInstance**(*context*).getString(KEY\_EMAIL, ""),  
 **getInstance**(*context*).getString(KEY\_PASSWORD, ""));  
 }  
 return null;  
}  
  
public static void saveSession(Context *context*, User *user*) {  
 **getInstance**(*context*).edit().putLong(KEY\_ID, *user*.getId()).apply();  
 **getInstance**(*context*).edit().putString(KEY\_EMAIL, *user*.getEmail()).apply();  
 **getInstance**(*context*).edit().putString(KEY\_PASSWORD, *user*.getPassword()).apply();  
}  
  
public static void clearSession(Context *context*) {  
 **getInstance**(*context*).edit().clear().apply();  
}

* Através deste codigo podemos guardar as sessões de um utilizador evitando que o mesmo esteja sempre a colocar os seus dados.

# Partes importantes do código

public class GameRepository {  
  
 private final Context context;  
 private GameDao gameDao;  
 private ReviewDao gameWithReview;  
  
 public GameRepository(Context *context*){  
 this.gameDao = AppDatabase.**getInstance**(*context*).getGameDao();  
 this.gameWithReview = AppDatabase.**getInstance**(*context*).getReviewDao();  
 this.context = *context*;  
 }  
  
 public LiveData<List<GameWithReview>> getGames() {  
 this.updateGames();  
 return this.gameDao.getAllGames();  
 }  
  
 public LiveData<Game> getGameId(long *id*) {  
 this.updateGames();  
 return this.gameDao.getGame(*id*);  
 }  
  
 public void updateGames() {  
 GameService gameService = DataSource.**getGameService**();  
 Call<List<Game>> call = gameService.getGames();  
  
 call.enqueue(new Callback<List<Game>>() {  
 *@Override* public void onResponse(Call<List<Game>> *call*, Response<List<Game>> *response*) {  
 if(*response*.isSuccessful()){  
 List<Game> gameList = *response*.body();  
  
 new Thread(new Runnable() {  
 *@Override* public void run() {  
 gameDao.addGames(gameList);  
 }  
 }).start();  
 }  
 }  
  
 *@Override* public void onFailure(Call<List<Game>> *call*, Throwable *t*) {  
 *t*.printStackTrace();  
 }  
 });  
 }  
  
 public LiveData<List<GameCart>> getGamesInCart(long *id*) {  
 GameService gameService = DataSource.**getGameService**();  
 Call<List<GameCart>> call = gameService.getGamesInCart(*id*);  
  
 call.enqueue(new Callback<List<GameCart>>() {  
 *@Override* public void onResponse(Call<List<GameCart>> *call*, Response<List<GameCart>> *response*) {  
 if(*response*.isSuccessful()){  
 new Thread(new Runnable() {  
 *@Override* public void run() {  
 gameDao.addGameToCart(*response*.body());  
 }  
 }).start();  
 }  
 }  
  
 *@Override* public void onFailure(Call<List<GameCart>> *call*, Throwable *t*) {  
 *t*.printStackTrace();  
 }  
 });  
  
 return gameDao.getAllInCart();  
 }  
  
 public void createUser(User *user*) {  
 GameService service = DataSource.**getGameService**();  
 service.addUser(*user*).enqueue(new Callback<User>() {  
 *@Override* public void onResponse(Call<User> *call*, Response<User> *response*) {  
 User users = *response*.body();  
 }  
  
 *@Override* public void onFailure(Call<User> *call*, Throwable *t*) {  
 *t*.printStackTrace();  
 }  
 });  
 }  
  
 public LiveData<User> getUserByEmail(Context *context*, String *email*) {  
 MutableLiveData<User> userMutableLiveData = new MutableLiveData<>();  
 GameService service = DataSource.**getGameService**();  
 service.getUserByEmail(*email*).enqueue(new Callback<List<User>>() {  
 *@Override* public void onResponse(Call<List<User>> *call*, Response<List<User>> *response*) {  
 if (*response*.isSuccessful()){  
 List<User> users = *response*.body();  
 if (users.size() > 0) {  
 userMutableLiveData.postValue(*response*.body().get(0));  
 }else {  
 userMutableLiveData.postValue(null);  
 Toast toast = Toast.**makeText**(*context*, "Email já existe",Toast.LENGTH\_SHORT);  
 toast.show();  
 }  
 }else {  
 userMutableLiveData.postValue(null);  
 }  
 }  
 *@Override* public void onFailure(Call<List<User>> *call*, Throwable *t*) {  
 *t*.printStackTrace();  
 userMutableLiveData.postValue(null);  
 }  
 });  
 return userMutableLiveData;  
 }  
  
 public LiveData<UserResponse> getUserByPasswordAndEmail(Context *context*, String *email*, String *password*) {  
 MutableLiveData<UserResponse> userMutableLiveData = new MutableLiveData<>();  
 GameService service = DataSource.**getGameService**();  
 service.getUserByEmailAndPassword(new User(0, *password*, *email*)).enqueue(new Callback<UserResponse>() {  
 *@Override* public void onResponse(Call<UserResponse> *call*, Response<UserResponse> *response*) {  
 UserResponse users = *response*.body();  
 if (users != null) {  
 userMutableLiveData.postValue(*response*.body());  
 }  
 }  
  
 *@Override* public void onFailure(Call<UserResponse> *call*, Throwable *t*) {  
 userMutableLiveData.postValue(null);  
 }  
 });  
  
 return userMutableLiveData;  
 }  
}

É com este código que podemos utilizar a nossa API criada em Laravel.

# Partes importantes do código

*@Entity*public class Game {  
  
 *@PrimaryKey*(autoGenerate = true)  
 private long id;  
 private String imgURL;  
 private String title;  
 private String description;  
 private String category;  
 private String console;  
 private float price;  
 private boolean isInCart;  
  
 public Game(long *id*, String *imgURL*, String *title*, String *description*, String *category*, String *console*, float *price*, boolean *isInCart*) {  
 this.id = *id*;  
 this.imgURL = *imgURL*;  
 this.title = *title*;  
 this.description = *description*;  
 this.category = *category*;  
 this.console = *console*;  
 this.price = *price*;  
 this.isInCart = *isInCart*;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long *id*) {  
 this.id = *id*;  
 }  
  
 public String getImgURL() {  
 return imgURL;  
 }  
  
 public void setImgURL(String *imgURL*) {  
 this.imgURL = *imgURL*;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String *title*) {  
 this.title = *title*;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String *description*) {  
 this.description = *description*;  
 }  
  
 public String getCategory() {  
 return category;  
 }  
  
 public void setCategory(String *category*) {  
 this.category = *category*;  
 }  
  
 public String getConsole() {  
 return console;  
 }  
  
 public void setConsole(String *console*) {  
 this.console = *console*;  
 }  
  
 public float getPrice() {  
 return price;  
 }  
  
 public void setPrice(float *price*) {  
 this.price = *price*;  
 }  
  
 public boolean isInCart() {  
 return isInCart;  
 }  
  
 public void setInCart(boolean *inCart*) {  
 isInCart = *inCart*;  
 }  
}

Este é o codigo do nosso objeto principal

# Conclusão

Neste projeto acabamos por fazer uma aplicação que oferece aos utilizadores uma forma de requisitar livros disponíveis.

Cumprimos todos os requisitos que nos foram propostos, e tivemos um excelente trabalho em dupla.

Este projeto foi de grande importância para nós, pois fez-nos juntar tudo o que aprendemos durante o semestre nas disciplinas, Programação de dispositivos móveis e Sistemas interativos.

Concluímos assim o nosso projeto com excelentes bases nas linguagens Java com android Studio e com grande vontade para continuar a aprender mais.



Figura 14- Stickman a certificar tarefas

# Referências Bibliográficas

<https://developer.android.com/studio?gclid=Cj0KCQjw5auGBhDEARIsAFyNm9FLk8GasacIcEmpeQDMiR1jAXPgssrli4Oe-r8ypbfX6GTFMgNLRhoaApBeEALw_wcB&gclsrc=aw.ds>

https://central.github.com/deployments/desktop/desktop/latest/win32

<https://adobe-xd.softonic.com.br/download>

<https://www.adobe.com/pt/products/illustrator/free-trial-download.html#mini-plans-web-cta-illustrator-card>

Exercícios lecionados nas aulas de sistemas interativos

Exercícios lecionados nas aulas de programação para dispositivos móveis

<https://developer.android.com>

<https://pt.stackoverflow.com>