

**UNIVERSIDADE NOVA DE LISBOA**

Faculdade de Ciências e Tecnologia

Departamento de Informática

# **Relatório da 2<sup>a</sup> Fase Projeto Base de Dados**



## **PLATAFORMA DE JOGOS**

G107

André Hindi Burrows nº 62184

Lucas Deda de Andrade nº 64583

Nícolas Carlos Nascimento nº 61905

Turno Prático dos Alunos: P2

# Plataforma de Distribuição Digital de Jogos

## Descrição

Trata-se de uma base de dados para uma plataforma online/offline que oferece a gestão de uma biblioteca de jogos digitais com uma loja integrada e ferramentas sociais. É possível navegar pela plataforma para obter diversas informações sobre cada jogo, utilizadores e empresas a qualquer momento.

A plataforma permite um registo onde se identifica o tipo de utilizador, jogador e/ou desenvolvedor, disponibilizando ferramentas diferentes dependendo do seu tipo. No geral, os utilizadores podem, além de gerir a sua biblioteca de jogos, escrever uma avaliação do jogo para a plataforma e criar servidores de jogos multijogador.

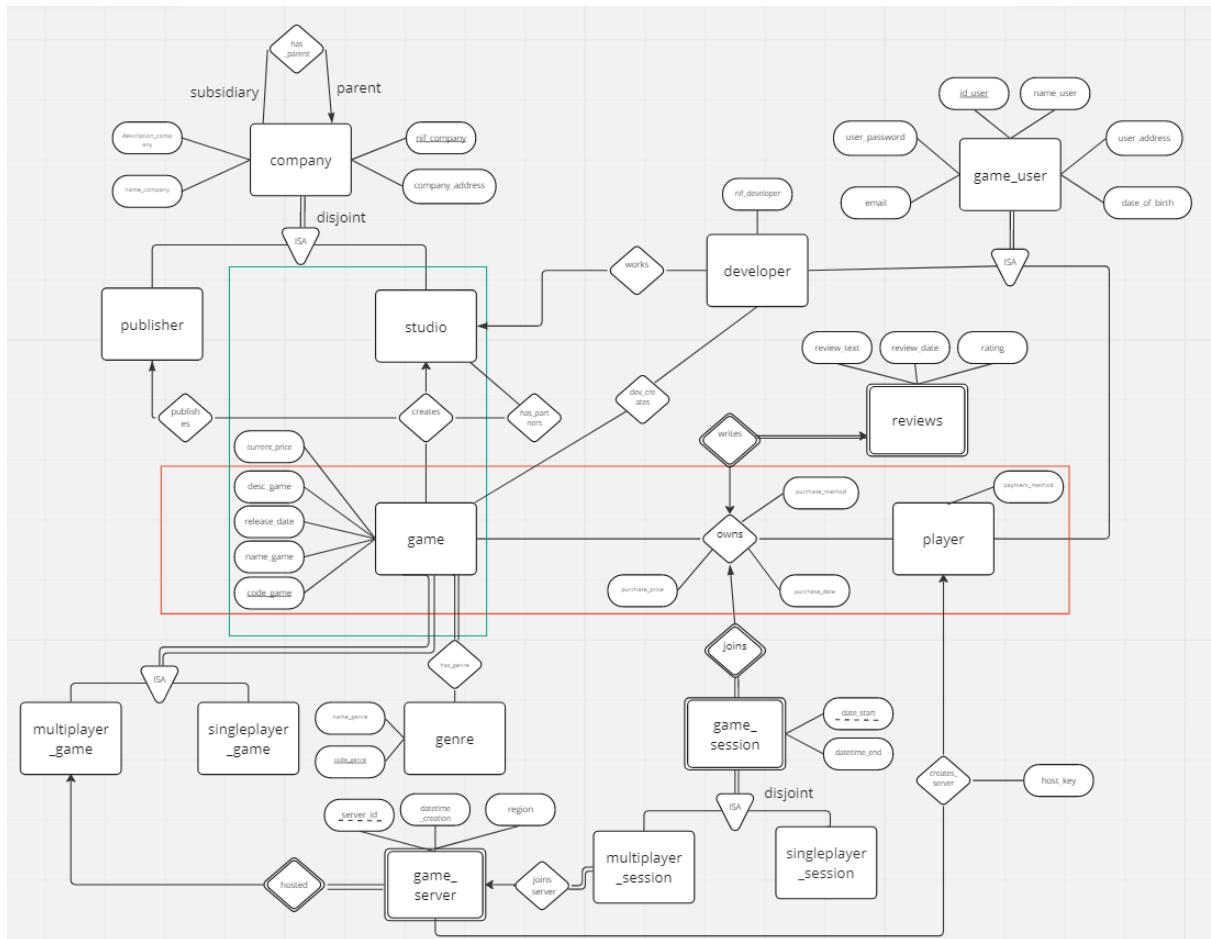
Com isto posto, a parte social consiste em possibilitar uma interação de todos os tipos de utilizadores, através de críticas e sessões multijogador. Um jogador regista uma avaliação de um jogo apenas uma vez, esta crítica contém um possível texto, uma classificação (de 1 a 5) e a data de envio da crítica. As sessões multijogador unem diversos jogadores num servidor de um jogo multijogador, havendo a possibilidade dos jogadores criarem servidores nestes jogos quando têm permissão.

Um jogo na plataforma pode ser criado de duas formas diferentes: 1. por um grupo de desenvolvedores (um ou mais) sem nenhum estúdio associado, ou 2. por um estúdio (onde trabalham desenvolvedores), podendo também ter o auxílio de outros estúdios (nesta situação chamados de parceiros). Esses jogos podem ou não ter distribuição por uma distribuidora. Estúdios e distribuidoras podem ser subsidiárias de uma empresa maior. Todos os jogos têm um nome único e são definidos por um ou mais géneros (por exemplo, se são de ação, são RPGs, etc.).

Mais detalhes serão expostos em restrições e comentados nas componentes do respetivo modelo a seguir.

# Modelo ER

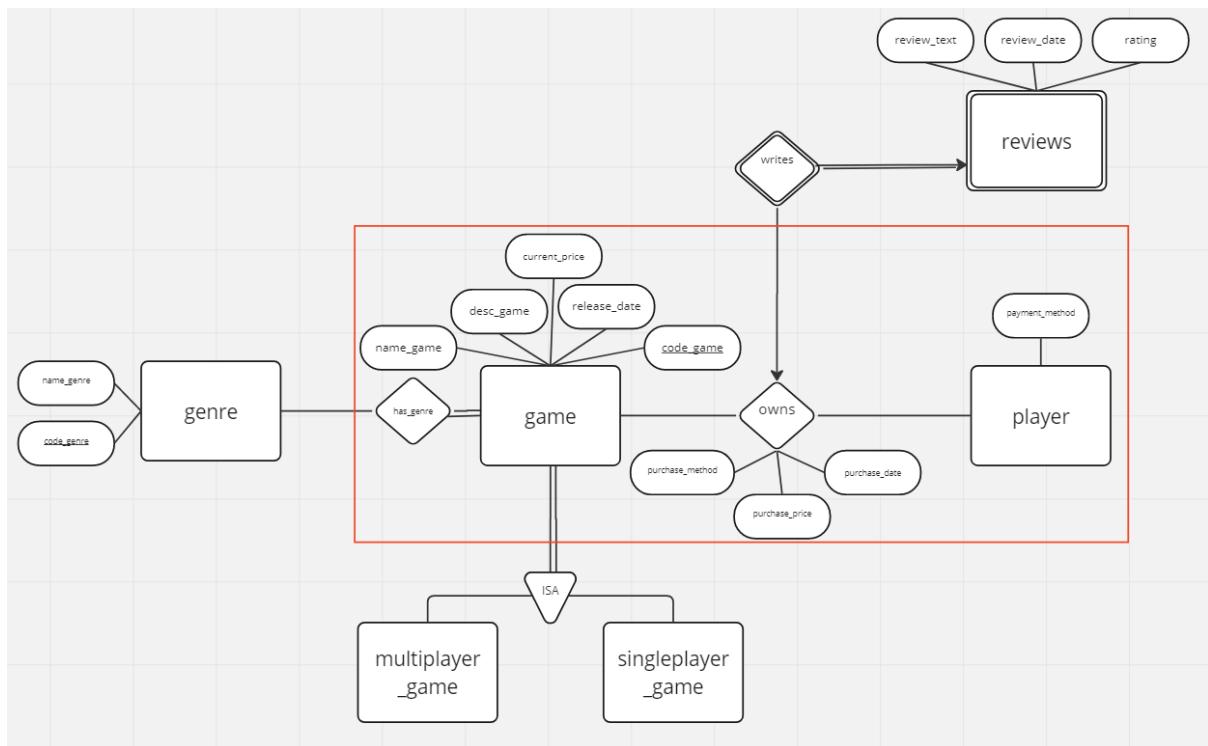
## Modelo completo da base de dados



Modelo completo da base de dados

Para facilitar a compreensão do modelo, decidimos descrever mais detalhadamente cada uma das suas componentes abaixo. Entretanto, é importante lembrar que **estas componentes não podem ser interpretadas isoladamente das outras**.

## Componente *owns*



## Componente *owns*

Armazenamos a informação dos jogos na entidade "game", utilizando um código de identificação sequencialmente atribuído por ordem de entrada na plataforma como chave primária (`code_game`). Os outros atributos para o jogo são: Nome (`name_game`), descrição (`desc_game`), preço atual (`current_price`) e data de lançamento (`release_date`). De forma a facilitar a implementação das sessões multi-jogador e dos servidores destes jogos (discutidos mais à frente na componente sessões-servidores), os jogos são obrigatoriamente especializados em jogo de jogador único (`singleplayer_game`) ou multijogador (`multiplayer_game`), podendo ser ambos.

A compra de jogos pelos jogadores é representada pela agregação "owns". Nela também armazenamos a data da compra (`purchase_date`), o preço pago nesta altura (`purchase_price`) e o método de pagamento utilizado (`purchase_method`). Nota: os jogadores, assim como os demais utilizadores do sistema, serão discutidos mais à frente na componente utilizadores.

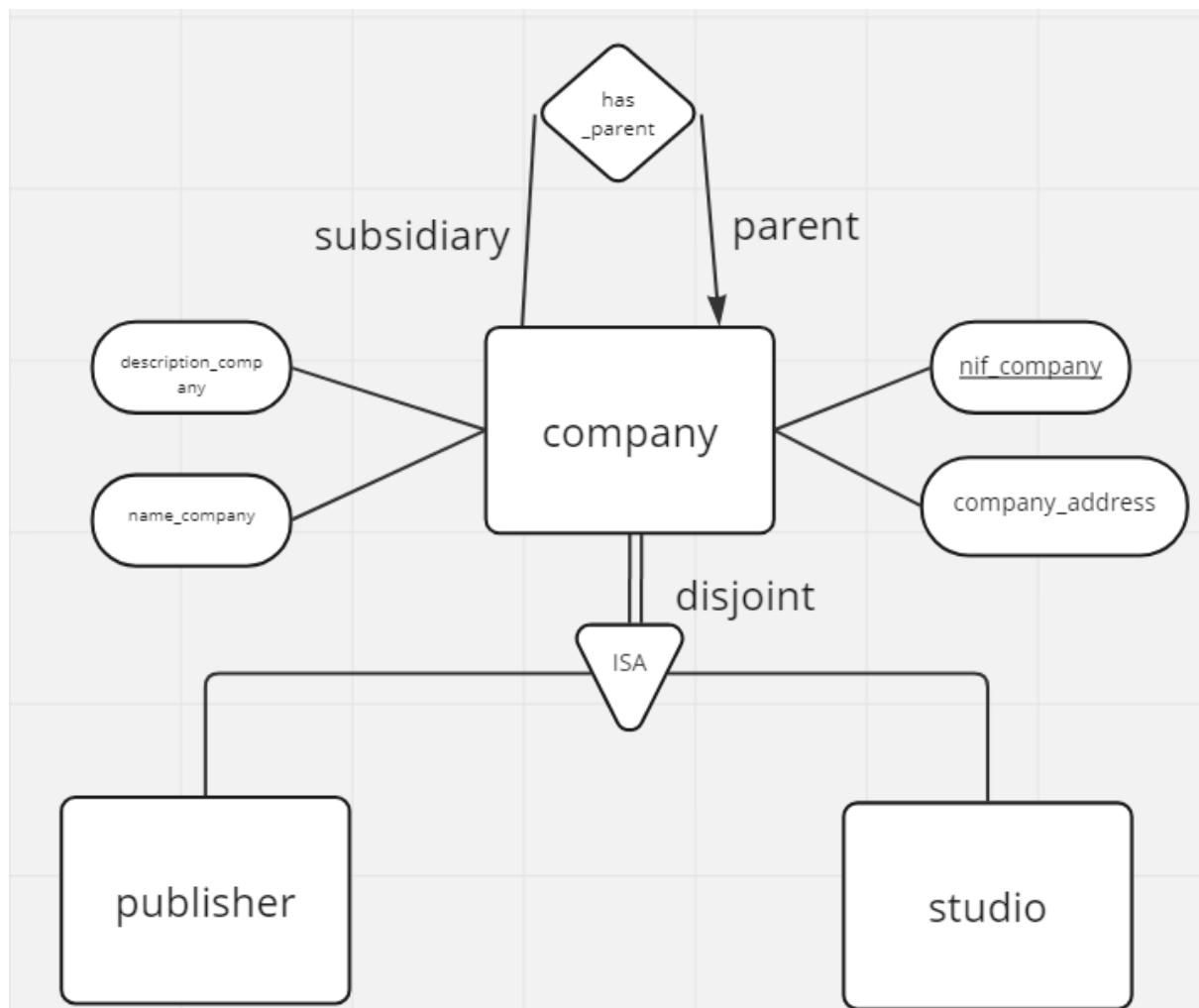
Todos os jogos se encaixam em pelo menos um género, sendo estes representados pelo conjunto de entidades **genre**. Cada género tem um código (`code_genre`) como chave primária, e também é armazenada o seu nome (`name_genre`).

Uma parte importante da plataforma são as avaliações (**reviews**), que são escritas por jogadores que possuem o jogo a ser avaliado. Uma review tem uma nota de 1 a 5 (`rating`)

dada pelo jogador, um texto opcional (*review\_text*), assim como uma data de publicação (*review\_date*).

Uma avaliação é unicamente identificada pelo jogo a qual se refere e o jogador deste jogo que está a escrevê-la. Portanto, é uma entidade fraca em relação ao conjunto de agregação *owns*. Como só se pode haver uma avaliação (que eventualmente poderá ser atualizada) de um jogador a um jogo que possui, não há discriminante para esta entidade.

## Componente das empresas



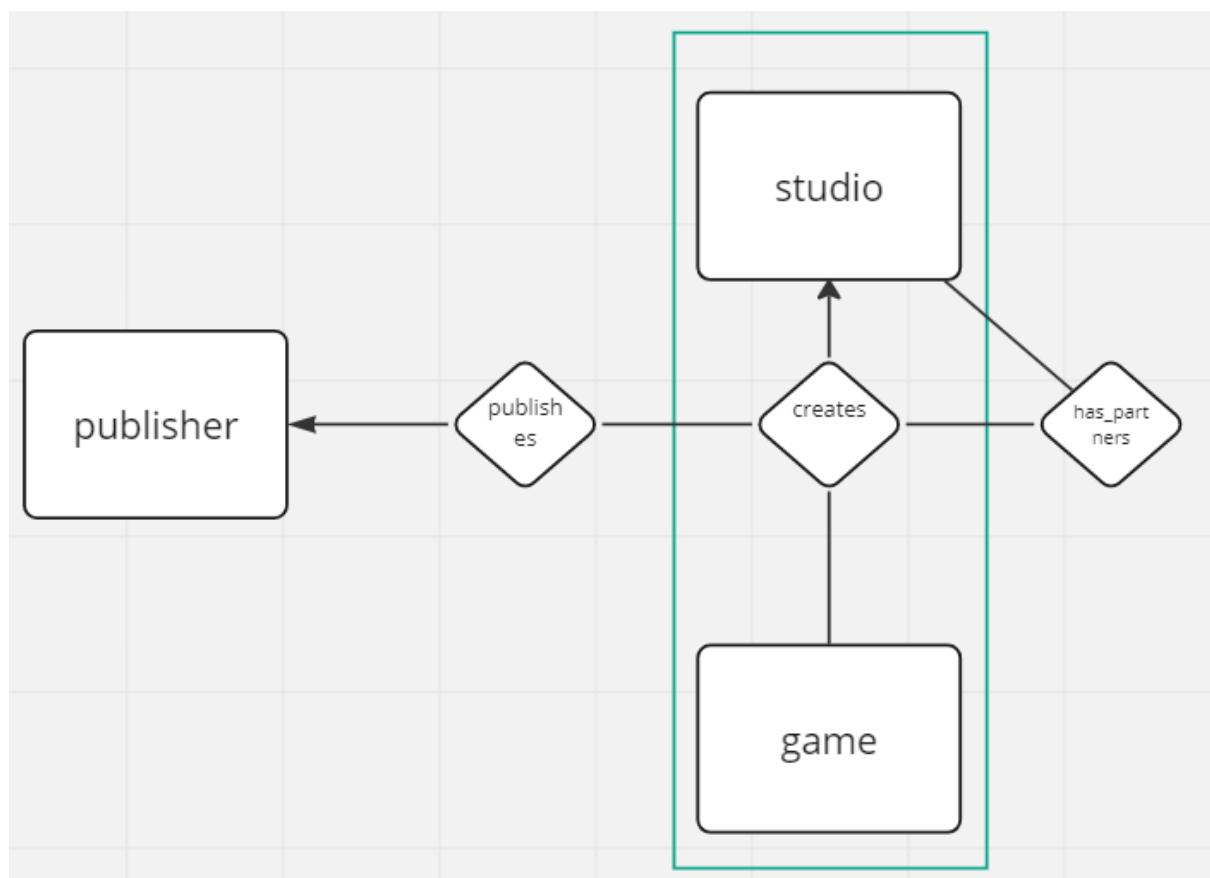
Componente das empresas

Para descrever uma empresa (*company*), selecionamos os seguintes atributos principais: descrição (*description\_company*), nome (*name\_company*), endereço (*company\_address*) e número de identificação fiscal (*nif\_company*) utilizado como chave primária da empresa, pelo facto de ser único.

Uma empresa deve ser obrigatoriamente uma distribuidora (*publisher*) ou um estúdio de desenvolvimento de jogos (*studio*), não podendo ser ambos. Embora compartilhem exatamente os mesmos atributos, queremos diferenciá-los, uma vez que desempenham papéis diferentes. Desta forma também conseguimos evitar que um desenvolvedor possa trabalhar numa distribuidora.

Como uma empresa pode ser “mãe” de várias distribuidoras e estúdios de desenvolvimento, que são neste caso consideradas subsidiárias, decidimos criar uma relação entre uma empresa e outra (com os papéis de subsidiária (*subsidiary*) e mãe (*parent*), respetivamente) para descrever esta situação.

## Componente da criação de um jogo por um estúdio



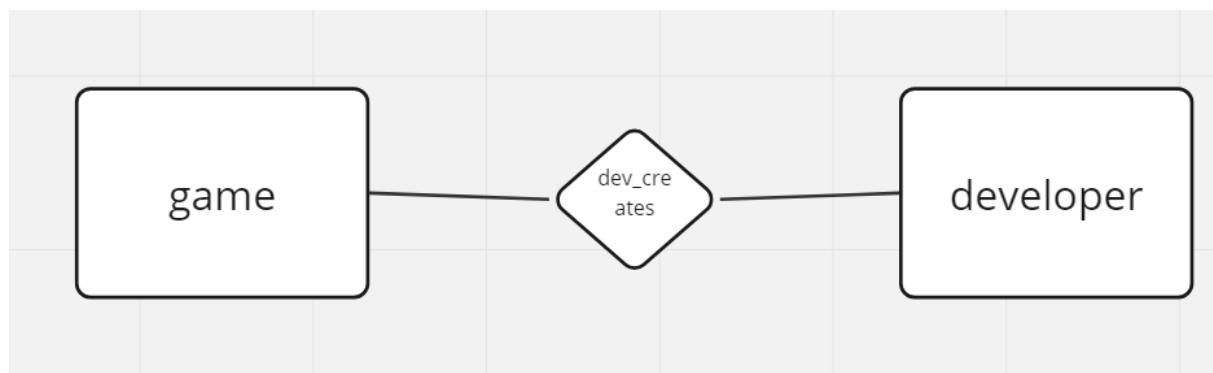
Componente da criação de um jogo por um estúdio

Um jogo pode ser desenvolvido por um ou mais estúdios, mas é necessário atribuir um e só um estúdio considerado o principal (via relação *creates*) quando este é o caso. Vários estúdios podem colaborar na criação de um ou mais jogos, desde que não sejam o estúdio principal responsável pelo jogo em causa.

É necessário notar que os jogos também podem ser criados sem a necessidade de que sejam atribuídos estúdios, sendo neste caso criados apenas por desenvolvedores (*developers*). Isto será explorado na componente criação de um jogo por desenvolvedores.

Quanto à distribuição dos jogos, há duas possibilidades: 1. O jogo pode ser independente e não ter uma distribuidora; 2. O jogo tem uma e só uma distribuidora. Associamos o jogo criado por um estúdio à distribuidora via a relação *publishes*.

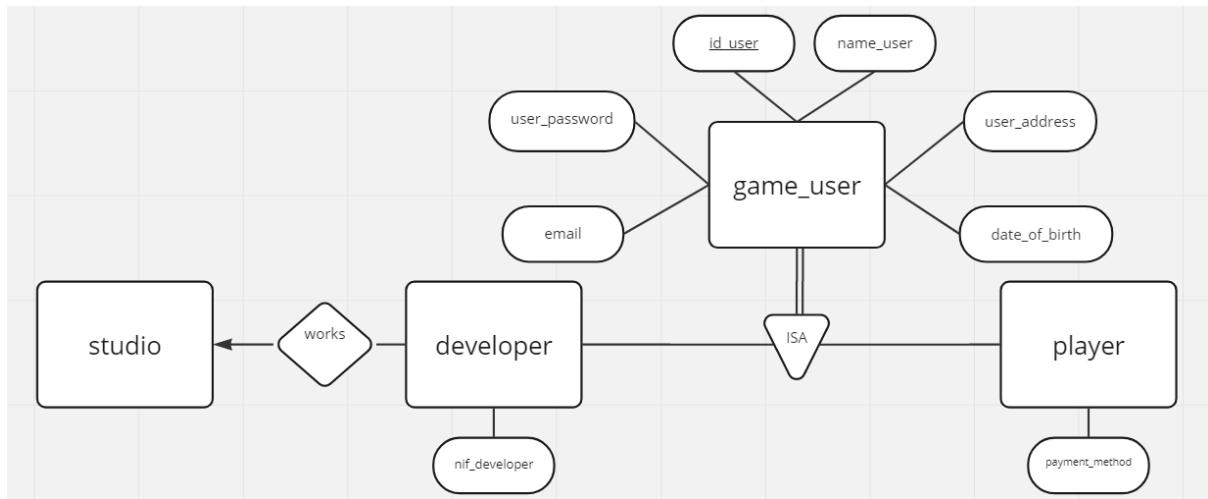
## Componente da criação de um jogo por desenvolvedores



Componente da criação de um jogo por desenvolvedores

Um jogo pode ser desenvolvido por um ou mais desenvolvedores. No entanto, quando um jogo é desenvolvido por desenvolvedores independentes, não pode ser simultaneamente distribuído por uma distribuidora ou criado por um estúdio. É importante destacar que a situação em que um desenvolvedor trabalha em um estúdio que cria um jogo (*works*) **não deve ser confundida** com a criação de jogos por desenvolvedores independentes (*dev\_creates*), pois, neste último caso, nenhum estúdio está envolvido.

## Componente dos utilizadores



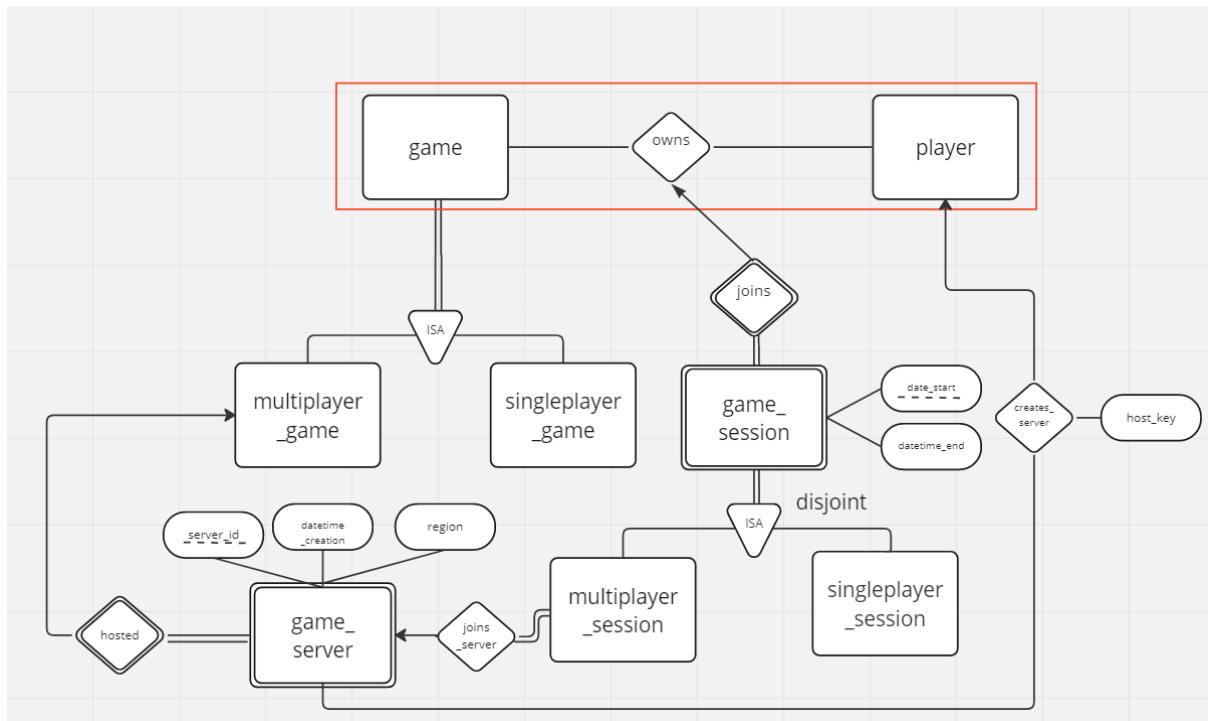
*Componente dos Utilizadores*

É necessário armazenar informação tanto sobre os desenvolvedores que criam jogos, quanto os jogadores que os jogam. Como a informação principal sobre ambos é muito similar, criamos a entidade utilizador (*game\_user*). Todos os utilizadores são unicamente identificados pelo seu Id (*id\_user*), e têm nome (*name\_user*), senha (*user\_password*), morada (*user\_address*), data de nascimento (*date\_of\_birth*) e endereço de email (*email*). Todos os utilizadores devem ser um desenvolvedor ou um jogador, podendo, no entanto, ser ambos.

Para os jogadores, é necessário armazenar o método de pagamento que usam para comprar jogos (*payment\_method*), e para os desenvolvedores armazena-se o seu número de contribuinte (*nif\_developer*).

Um desenvolvedor pode trabalhar (não obrigatoriamente) em um e um só estúdio de desenvolvimento de jogos.

## Componente sessões-servidores



### Componente sessões-servidores

Na nossa base de dados também queremos armazenar informação relativa ao tempo que um jogador dedica a um determinado jogo, e com quem jogou (caso tenha entrado em uma sessão multijogador). Surgiu então a entidade sessão (*session*), que é associada a um jogo e um jogador (via relação *owns*). Como depende desta relação para ser distinguida, então é fraca em relação a ela.

Consideramos que um jogador não pode jogar mais de um jogo da sua biblioteca ao mesmo tempo. Portanto, para diferenciar as sessões de um par jogo-jogador basta verificar a hora de início da sessão (*datetime\_start*). Como queremos poder calcular o tempo gasto em cada sessão, também armazenamos a data do fim da sessão (*datetime\_end*).

Uma sessão deve ser da modalidade de um único jogador (*singleplayer\_session*) ou multijogador (*multiplayer\_session*), obrigatoriamente. Logo temos uma especialização obrigatória e disjunta. O que diferencia uma sessão multijogador de uma *singleplayer* é o facto de que está relacionada ao servidor (*game\_server*) a qual o jogador se juntou.

O servidor é uma entidade que permite simular uma sessão multijogador. Ele é uma entidade fraca, tendo o seu jogo como entidade forte. Além disso, diferenciam-se servidores de um mesmo jogo com um código sequencialmente atribuído na criação do servidor (*server\_id*) como discriminante. Guardamos também a região do servidor (por exemplo, 'EU', 'US', etc.), assim como a data (*datetime\_creation*) na qual foi criado. Como o servidor

é uma entidade fraca, depende do código do jogo ao qual pertence. O servidor está sempre associado a um jogo multiplayer que o hospeda.

A criação de servidores pode ser feita de duas maneiras: 1. Por um jogador ou 2. Independentemente de um jogador. Jogadores que criam um servidor de um jogo inserem uma chave de hospedagem (*host\_key*), garantindo a permissão para tal ação. Desta maneira, um jogador não precisa adquirir um jogo para criar um servidor dele.

# Modelo Relacional

Nota: Chaves primárias são denotadas por sublinhado.

Company(nif\_company, name\_company, description\_company, address);

Publisher(nif\_company)

- nif\_company é uma chave estrangeira referenciando a relação Company.

Studio(nif\_company)

- nif\_company é uma chave estrangeira referenciando a relação Company.

Game\_User(id\_user, name\_user, email, user\_password, date\_of\_birth, user\_address);

Developer(id\_user, nif\_developer);

- id\_user é uma chave estrangeira referenciando a relação Game\_User.

Player(id\_user, payment\_method);

- id\_user é uma chave estrangeira referenciando a relação Game\_User.

Game(code\_game, name\_game, desc\_game, release\_date, current\_price);

Singleplayer\_Game(code\_game)

- code\_game é uma chave estrangeira referenciando a relação Game.

Multiplayer\_Game(code\_game)

- code\_game é uma chave estrangeira referenciando a relação Game.

Genre(code\_genre, name\_genre);

Owns(code\_game, id\_user, purchase\_price, purchase\_date, purchase\_method);

- code\_game é chave estrangeira referenciando a relação Game
- id\_user é uma chave estrangeira referenciando a relação Player

Reviews(id\_user, code\_game, review\_date, rating, review\_text);

- (id\_user, code\_game) é chave estrangeira referenciando a relação owns

Game\_Server(code\_game, server\_id, datetime\_creation, region);

- code\_game é uma chave estrangeira referenciando a relação Multiplayer\_Game.

Game\_Session(id\_user, code\_game, datetime\_start, datetime\_end)

- (id\_user, code\_game) é chave estrangeira referenciando a relação owns

Singleplayer\_session(id\_user, code\_game, datetime\_start);

- (id\_user, code\_game, datetime\_start) é chave estrangeira referenciando a relação Game\_Session

Multiplayer\_session(id\_user, code\_game, datetime\_start, s\_code\_game, server\_id);

- (id\_user, code\_game, datetime\_start) é chave estrangeira referenciando a relação Game\_Session
- (s\_code\_game, server\_id) é chave estrangeira referenciando a relação Game\_Server

has\_parent(nif\_subsidiary, nif\_parent)

- nif\_subsidiary é uma renomeação de nif\_company, chave estrangeira referenciando a relação Company
- nif\_parent é uma renomeação de nif\_company, chave estrangeira referenciando a relação Company

Creates(code\_game, nif\_studio);

- code\_game é uma chave estrangeira referenciando a relação Game.
- nif\_studio é uma renomeação de nif\_company, chave estrangeira de Studio

publishes(code\_game, nif\_publisher);

- nif\_publisher é uma renomeação de nif\_company, chave estrangeira de Publisher
- code\_game é chave estrangeira referenciando a relação Creates

works(id\_user, nif\_company);

- nif\_company é chave estrangeira referenciando a relação Studio
- id\_user é uma chave estrangeira referenciando a relação Developer.

devCreates(code\_game, id\_user);

- code\_game é chave estrangeira referenciando a relação Game
- id\_user é uma chave estrangeira referenciando a relação Developer

has\_genre(code\_game, code\_genre);

- code\_game é chave estrangeira referenciando a relação Game
- code\_genre é chave estrangeira referenciando a relação Genre

creates\_server(code\_game, server\_id, id\_user, host\_key);

- id\_user é uma chave estrangeira referenciando a relação Player
- (code\_game, server\_id) é chave estrangeira referenciando a relação Game\_Server

has\_partners(code\_game, nif\_company)

- code\_game é chave estrangeira referenciando a relação Game.
- nif\_company é chave estrangeira referenciando a relação Studio.

# Diferenças em relação à 1<sup>a</sup> fase

## Modelo ER

- Renomeou-se a entidade ‘user’ para *Game\_User*, uma vez que a palavra user é reservada em SQL.
  - Foram renomeados também os atributos *password* e *address* por *user\_password* e *user\_address* pelo mesmo motivo.
- O atributo *address* em *company* foi renomeado para *company\_address* por motivos de clareza.
- Foi adicionado o atributo *date\_of\_birth* em *Game\_User*.
- Review date deixa de ser um atributo discriminante de reviews, uma vez que optamos por não permitir mais de uma avaliação por agregado jogo/jogador.
- Foi adicionado o atributo *purchase\_method* na relação *owns* para permitir a armazenagem do método de compra utilizado na altura da compra de um determinado jogo por um jogador.
- Foram adicionados os atributos *name\_game* e *current\_price* na entidade *game* para armazenar o nome e o preço atual de um jogo no sistema, respetivamente.
- Por meio da nova relação *has\_partners*, estúdios podem colaborar em um jogo criado por outro estúdio.
- O jogo foi especializado em dois tipos, singleplayer e multiplayer.
- Eliminou-se o atributo da descrição de um género (*description\_genre*) na tabela *Genre*, sendo este substituído por *name\_genre*, o nome do género. Isto se deve ao facto de não acharmos a descrição do género relevante, sendo o seu nome de maior importância.
- Houve uma reestruturação da entidade *Game\_Server*. Ela passou a ser uma entidade fraca em relação aos jogos multiplayer, sendo diferenciada pelo seu id (*server\_id*) entre os servidores de um jogo em específico. A chave de hospedagem (*host\_key*) deixou de ser um atributo relacionado ao *Game\_Server* (erroneamente atribuído à relação *hosted*), mas sim da relação de criação de um servidor por um utilizador (*creates\_server*), onde faz sentido a sua presença.

## Modelo Relacional

- Foram corrigidos os erros em que o discriminante de uma entidade fraca estava sublinhado a tracejado, uma vez que isso não faz parte do modelo relacional.
- Por uma questão de clareza, foi removida a distinção no texto entre relações que representam entidades e outras relações, uma vez que no modelo relacional todas são relações.
- Os nomes dos atributos e relações foram atualizados de acordo com o que foi mencionado na seção anterior.
- Foram adicionados os atributos que não existiam anteriormente em suas respectivas tabelas, conforme mencionado na seção anterior.

- Foram adicionadas as seguintes relações: Singleplayer\_game, Multiplayer\_game, Has\_partners e Game\_session.
  - Anteriormente, a relação Game\_Session não estava presente. No entanto, notamos a necessidade de referenciar sessões em geral dentro da relação "joins", o que impedia "esconder" essa relação.
  - Em função disso, o atributo datetime\_end foi removido das tabelas Singleplayer\_session e Multiplayer\_session.
- Na tabela Reviews, a chave primária foi corrigida para ser uma chave estrangeira referente à relação Owns, e não às relações Game e Player separadamente.
- Na tabela Creates\_Server, a chave primária foi corrigida para ser uma chave estrangeira referente à relação Game\_Server, não contendo o código referente ao jogador, uma vez que um servidor só pode ser criado por um único jogador.
- A chave primária da tabela Game\_Server foi alterada para ser um par código de jogo (code\_game) e ID (server\_id), uma vez que é uma entidade fraca em relação a um jogo multiplayer e é distinta no mesmo jogo em relação ao seu ID.
- Na tabela Has\_Parent, a chave primária não contém mais o atributo nif\_parent, uma vez que uma subsidiária pode ter apenas uma empresa mãe.
- Na tabela Creates, a chave primária não contém mais o atributo nif\_studio, uma vez que um jogo criado por um estúdio pode ter apenas um estúdio principal (o presente na relação).
- Na tabela Publishes, a chave primária não contém mais o atributo nif\_studio, uma vez que, ao fazer referência à relação Creates, é necessário apenas o código do jogo (code\_game).
- Na tabela Works, a chave primária não contém mais o atributo nif\_company, uma vez que um desenvolvedor pode trabalhar no máximo em um único estúdio.

## Código SQL

### - Criação de Tabelas

No início de nosso código SQL, para além de excluir as tabelas (*drop table*), criamos todas as tabelas que devem existir de acordo com nosso Modelo Relacional. Elas são criadas da seguinte forma e com a seguinte especificação:

```
CREATE TABLE COMPANY (
    NIF_COMPANY VARCHAR2(9),
    NAME_COMPANY VARCHAR2(30) NOT NULL,
    DESCRIPTION_COMPANY VARCHAR2(100) NOT NULL,
    COMPANY_ADDRESS VARCHAR2(50) NOT NULL,
    PRIMARY KEY(NIF_COMPANY)
);
```

*Criação da tabela Company.*

```
CREATE TABLE PUBLISHER (
    NIF_COMPANY VARCHAR2(9),
    FOREIGN KEY (NIF_COMPANY) REFERENCES COMPANY(NIF_COMPANY),
    PRIMARY KEY (NIF_COMPANY)
);
```

*Criação da tabela Publisher.*

```
CREATE TABLE STUDIO (
    NIF_COMPANY VARCHAR2(9),
    FOREIGN KEY (NIF_COMPANY) REFERENCES COMPANY(NIF_COMPANY),
    PRIMARY KEY (NIF_COMPANY)
);
```

*Criação da tabela Studio.*

```
CREATE TABLE GAME_USER (
    ID_USER VARCHAR2(30),
    NAME_USER VARCHAR2(200) NOT NULL,
    EMAIL VARCHAR2(200) NOT NULL,
    USER_PASSWORD VARCHAR2(200) NOT NULL,
    DATE_OF_BIRTH TIMESTAMP NOT NULL,
    USER_ADDRESS VARCHAR2(200) NOT NULL,
    PRIMARY KEY (ID_USER)
);
```

*Criação da tabela Game\_User.*

Um tipo *TIMESTAMP* é usado para armazenar a data de nascimento. Apesar de não ser necessário armazenar o horário, a escolha deste tipo é necessária para manter a compatibilidade com outras tabelas. Na aplicação é possível criar usuários, através de um procedimento que será demonstrado a seguir.

```
CREATE TABLE PLAYER (
    ID_USER VARCHAR2(30),
    PAYMENT_METHOD VARCHAR2(200) NOT NULL,
    FOREIGN KEY (ID_USER) REFERENCES GAME_USER(ID_USER),
    PRIMARY KEY (ID_USER)
);
```

*Criação da tabela Player.*

```
CREATE TABLE GAME (
    CODE_GAME INT,
    NAME_GAME VARCHAR2(80) NOT NULL,
    DESC_GAME VARCHAR2(500) NOT NULL,
    RELEASE_DATE DATE DEFAULT SYSDATE NOT NULL,
    CURRENT_PRICE DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (CODE_GAME)
);
```

*Criação da tabela Game.*

*Nesta única tabela não é necessário armazenar a hora, logo foi usado um tipo DATE. CODE\_GAME é um valor atribuído exclusivamente pelo procedimento responsável pela sua criação.*

```
CREATE TABLE SINGLEPLAYER_GAME (
    CODE_GAME INT,
    FOREIGN KEY(CODE_GAME) REFERENCES GAME(CODE_GAME) DEFERRABLE,
    PRIMARY KEY (CODE_GAME)
);
```

*Criação da tabela Singleplayer\_game. Uma constraint DEFERRABLE é usada para que a tabela, quando criada, permita que temporariamente não haja um CODE\_GAME associado, para imediatamente depois associá-lo.*

```
CREATE TABLE MULTIPLAYER_GAME (
    CODE_GAME INT,
    FOREIGN KEY(CODE_GAME) REFERENCES GAME(CODE_GAME) DEFERRABLE,
    PRIMARY KEY (CODE_GAME)
);
```

*Criação da tabela Multiplayer\_game. A mesma constraint DEFERRABLE é usada pelo mesmo motivo de Singleplayer\_game.*

```
CREATE TABLE GENRE (
    CODE_GENRE INT,
    NAME_GENRE VARCHAR2(30) NOT NULL,
    PRIMARY KEY (CODE_GENRE)
);
```

Criação da tabela Genre.

```
CREATE TABLE OWNS (
    CODE_GAME INT,
    ID_USER VARCHAR2(30),
    PURCHASE_PRICE DECIMAL(10, 2) NOT NULL,
    PURCHASE_DATE TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    PURCHASE_METHOD VARCHAR2(200) NOT NULL,
    FOREIGN KEY (CODE_GAME) REFERENCES GAME(CODE_GAME) DEFERRABLE,
    FOREIGN KEY (ID_USER) REFERENCES PLAYER(ID_USER),
    PRIMARY KEY (CODE_GAME, ID_USER)
);
```

Criação da tabela Owns. A data e hora da compra é armazenada no tipo `TIMESTAMP`, e caso não seja dada uma data e hora, o atributo é automaticamente a hora atual.

```
CREATE TABLE REVIEWS (
    ID_USER VARCHAR2(30),
    CODE_GAME INT,
    REVIEW_DATE TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    RATING INT NOT NULL,
    REVIEW_TEXT VARCHAR2(500),
    FOREIGN KEY (ID_USER, CODE_GAME) REFERENCES OWNS(ID_USER, CODE_GAME) DEFERRABLE,
    PRIMARY KEY (ID_USER, CODE_GAME),
    CHECK(RATING >= 1 AND RATING <= 5)
);
```

Criação da tabela Reviews. A data da criação da review tem a opção de ser definida como alguma hora específica, mas na maioria dos casos é usado o valor padrão, a data e hora atual. É utilizada a constraint `CHECK` para garantir que o valor de `RATING` seja um valor entre 1 e 5 inclusive.

```

CREATE TABLE GAME_SERVER (
    CODE_GAME INT,
    SERVER_ID INT,
    DATETIME_CREATION TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    REGION VARCHAR2(200) NOT NULL,
    FOREIGN KEY (CODE_GAME) REFERENCES MULTIPLAYER_GAME(CODE_GAME) DEFERRABLE,
    PRIMARY KEY (CODE_GAME, SERVER_ID)
);

```

*Criação da tabela Game\_server. A data e hora da criação é, por padrão, a hora atual. Server ID é um atributo exclusivamente atribuído pelo procedimento responsável pela criação de um servidor, ilustrado no próximo tópico.*

```

CREATE TABLE GAME_SESSION (
    ID_USER VARCHAR2(30),
    CODE_GAME INT,
    DATE_START TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    DATETIME_END TIMESTAMP,
    FOREIGN KEY (ID_USER, CODE_GAME) REFERENCES OWNS(ID_USER, CODE_GAME) DEFERRABLE,
    PRIMARY KEY (ID_USER, CODE_GAME, DATE_START)
);

```

*Criação da tabela Game\_session. A data e hora de início é a hora atual, mas não há padrão para a data e hora de término da sessão. Esta também pode ser nula, ao contrário da maioria dos atributos nesta base de dados, para indicar que uma sessão não foi terminada ainda.*

```

CREATE TABLE SINGLEPLAYER_SESSION (
    ID_USER VARCHAR2(30),
    CODE_GAME INT,
    DATE_START TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ID_USER, CODE_GAME, DATE_START) REFERENCES GAME_SESSION(ID_USER, CODE_GAME, DATE_START) DEFERRABLE,
    PRIMARY KEY (ID_USER, CODE_GAME, DATE_START)
);

```

*Criação da tabela Singleplayer\_session. A data de início referencia a tabela Game\_session, logo possui as mesmas propriedades.*

```

CREATE TABLE MULTIPLAYER_SESSION (
    ID_USER VARCHAR2(30),
    CODE_GAME INT,
    DATE_START TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    SERVER_ID INT NOT NULL,
    S_CODE_GAME INT NOT NULL,
    FOREIGN KEY (ID_USER, CODE_GAME, DATE_START) REFERENCES GAME_SESSION(ID_USER, CODE_GAME, DATE_START) DEFERRABLE,
    FOREIGN KEY (S_CODE_GAME, SERVER_ID) REFERENCES GAME_SERVER(CODE_GAME, SERVER_ID) DEFERRABLE,
    PRIMARY KEY (ID_USER, CODE_GAME, DATE_START),
    CHECK (CODE_GAME = S_CODE_GAME)
);

```

*Criação da tabela Multiplayer\_session. Similar à tabela Singleplayer\_session, a data de início tem as mesmas propriedades que a de Game\_session.*

```
CREATE TABLE HAS_PARENT (
    NIF_SUBSIDIARY VARCHAR2(9),
    NIF_PARENT VARCHAR2(9) NOT NULL,
    FOREIGN KEY (NIF_SUBSIDIARY) REFERENCES COMPANY(NIF_COMPANY),
    FOREIGN KEY (NIF_PARENT) REFERENCES COMPANY(NIF_COMPANY),
    PRIMARY KEY (NIF_SUBSIDIARY),
    CHECK(NIF_SUBSIDIARY <> NIF_PARENT)
);
```

*Criação da tabela Has\_parent. Há uma constraint CHECK para garantir que o NIF da empresa pai não é o mesmo da empresa filha, impedindo uma empresa de ser filial dela mesma.*

```
CREATE TABLE CREATES (
    CODE_GAME INT,
    NIF_STUDIO VARCHAR2(9) NOT NULL,
    FOREIGN KEY (CODE_GAME) REFERENCES GAME(CODE_GAME) DEFERRABLE,
    FOREIGN KEY (NIF_STUDIO) REFERENCES STUDIO(NIF_COMPANY),
    PRIMARY KEY (CODE_GAME)
);
```

*Criação da tabela Creates. CODE\_GAME possui a constraint DEFERRABLE para permitir a sua criação sem um GAME associado temporariamente, que logo depois, no mesmo procedimento, é definido.*

```
CREATE TABLE PUBLISHES (
    CODE_GAME INT,
    NIF_PUBLISHER VARCHAR2(9) NOT NULL,
    FOREIGN KEY (CODE_GAME) REFERENCES CREATES(CODE_GAME) DEFERRABLE,
    FOREIGN KEY (NIF_PUBLISHER) REFERENCES PUBLISHER(NIF_COMPANY),
    PRIMARY KEY (CODE_GAME)
);
```

*Criação da tabela Publishes. CODE\_GAME possui a constraint DEFERRABLE pelo mesmo motivo que a tabela Creates.*

```
CREATE TABLE WORKS (
    ID_USER VARCHAR2(30),
    NIF_COMPANY VARCHAR2(9) NOT NULL,
    FOREIGN KEY (ID_USER) REFERENCES DEVELOPER(ID_USER),
    FOREIGN KEY (NIF_COMPANY) REFERENCES STUDIO(NIF_COMPANY),
    PRIMARY KEY (ID_USER)
);
```

Criação da tabela Works.

```
CREATE TABLE DEV_CREATES (
    CODE_GAME INT,
    ID_USER VARCHAR2(30),
    FOREIGN KEY (CODE_GAME) REFERENCES GAME(CODE_GAME) DEFERRABLE,
    FOREIGN KEY (ID_USER) REFERENCES DEVELOPER(ID_USER),
    PRIMARY KEY (CODE_GAME, ID_USER)
);
```

Criação da tabela DevCreates. CODE\_GAME possui a constraint DEFERRABLE pelo mesmo motivo que a tabela Creates.

```
CREATE TABLE HAS_GENRE (
    CODE_GAME INT,
    CODE_GENRE INT,
    FOREIGN KEY (CODE_GAME) REFERENCES GAME(CODE_GAME) DEFERRABLE,
    FOREIGN KEY (CODE_GENRE) REFERENCES GENRE(CODE_GENRE),
    PRIMARY KEY (CODE_GAME, CODE_GENRE)
);
```

Criação da tabela HasGenre. CODE\_GAME possui a constraint DEFERRABLE pelo mesmo motivo que a tabela Creates.

```

CREATE TABLE CREATES_SERVER (
    CODE_GAME INT,
    SERVER_ID INT,
    ID_USER VARCHAR2(30),
    HOST_KEY VARCHAR2(200) NOT NULL,
    FOREIGN KEY (ID_USER) REFERENCES PLAYER(ID_USER),
    FOREIGN KEY (CODE_GAME, SERVER_ID) REFERENCES GAME_SERVER(CODE_GAME, SERVER_ID),
    PRIMARY KEY (CODE_GAME, SERVER_ID)
);

```

*Criação da tabela Creates\_server.*

```

CREATE TABLE HAS_PARTNERS (
    CODE_GAME INT,
    NIF_COMPANY VARCHAR2(9),
    FOREIGN KEY (CODE_GAME) REFERENCES CREATES(CODE_GAME) DEFERRABLE,
    FOREIGN KEY (NIF_COMPANY) REFERENCES STUDIO(NIF_COMPANY),
    PRIMARY KEY (CODE_GAME, NIF_COMPANY)
);

```

*Criação da tabela Has\_partners.* CODE\_GAME possui a constraint DEFERRABLE pelo mesmo motivo que a tabela Creates.

## Triggers, Funções e Procedimentos do Código SQL

Foi necessário criar Triggers para garantir que os dados inseridos estejam corretamente formatados e que não haja partes faltando ou informação inconsistente. Utilizamos funções auxiliares para facilitar a implementação dos procedimentos e triggers e em alguns casos fornecer dados importantes à aplicação. Os procedimentos, em sua maioria, estão sendo utilizados para automatizar a inserção de dados na tabela. Segue o respectivo código de cada um dos tipos:

```

CREATE OR REPLACE TRIGGER CHECK_GAME_HAS_GENRE BEFORE
| INSERT OR UPDATE ON GAME FOR EACH ROW
DECLARE
    GENRECOUNT INT;
BEGIN
    SELECT
        COUNT(*) INTO GENRECOUNT
    FROM
        HAS_GENRE
    WHERE
        CODE_GAME = :NEW.CODE_GAME;
    IF GENRECOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Game: All games must have at least one genre.');
    END IF;
END;
/

```

*Criação do trigger Check Game Has Genre.* É responsável por verificar, apenas quando a sessão é alterada para as constraints serem IMMEDIATE (graças a constraint DEFERRABLE na criação da tabela, ilustrada acima) (como é feito no procedimento Add Game), se um jogo possui ao menos um gênero associado a ele. Colocando um erro caso contrário.

```

CREATE OR REPLACE TRIGGER IS_NOT_ALREADY_STUDIO_CREATED BEFORE
| INSERT ON DEV_CREATES FOR EACH ROW
DECLARE
    CREATED_COUNT INT;
BEGIN
    SELECT
        COUNT(*) INTO CREATED_COUNT
    FROM
        CREATES
    WHERE
        CODE_GAME = :NEW.CODE_GAME;
    IF CREATED_COUNT <> 0 THEN
        RAISE_APPLICATION_ERROR (-20101, 'This game is already developed by a studio.');
    END IF;
END;
/

```

*Criação do trigger Is Not Already Studio Created.* Verifica, no registo de um jogo como criado somente por desenvolvedores (inserção na tabela DevCreates), se este já não foi criado por um estúdio (já está na tabela Creates). Caso esteja, é colocado um erro.

```

CREATE OR REPLACE TRIGGER IS_NOT_ALREADY_DEV_CREATED BEFORE
| INSERT ON CREATES FOR EACH ROW
DECLARE
    CREATED_COUNT INT;
BEGIN
    SELECT
        | COUNT(*) INTO CREATED_COUNT
    FROM
        | DEV_CREATES
    WHERE
        | CODE_GAME = :NEW.CODE_GAME;
    IF CREATED_COUNT <> 0 THEN
        RAISE_APPLICATION_ERROR (-20101, 'This game is already developed by an independent developer.');
    END IF;
END;
/

```

*Criação do trigger Is Not Already Dev Created. Sua função é um espelho de Is Not Already Studio Created. Responsável por garantir que no registo de um jogo como desenvolvido por um estúdio, este já não está registado como desenvolvido por desenvolvedores independentes. Colocando um erro quando este é o caso.*

```

CREATE OR REPLACE TRIGGER IS_NOT_MAIN_STUDIO BEFORE
| INSERT ON HAS_PARTNERS FOR EACH ROW
DECLARE
    NIF_MAIN_STUDIO VARCHAR2(9);
BEGIN
    SELECT
        | NIF_STUDIO INTO NIF_MAIN_STUDIO
    FROM
        | CREATES
    WHERE
        | CODE_GAME = :NEW.CODE_GAME;
    IF :NEW.NIF_COMPANY = NIF_MAIN_STUDIO THEN
        RAISE_APPLICATION_ERROR (-20101, 'A studio can''t be a partner to a game it is the main developer of.');
    END IF;
END;
/

```

*Criação do trigger Is Not Main Studio.*

*No registo de um estúdio como parceiro na criação de um jogo (inserção na tabela Has\_partners), garante que este estúdio já não é o principal. Caso contrário, coloca um erro.*

```

CREATE OR REPLACE FUNCTION COUNT_ACTIVE_SESSIONS (
    ID_USER VARCHAR2
) RETURN INT AS
    SESSION_COUNT INT;
BEGIN
    SELECT
        COUNT(*) INTO SESSION_COUNT
    FROM
        GAME_SESSION
    WHERE
        ID_USER = COUNT_ACTIVE_SESSIONS.ID_USER
        AND DATETIME_END IS NULL;
    RETURN SESSION_COUNT;
END;
/

```

*Criação da função Count Active Sessions. Recebe como parâmetro o ID do jogador e retorna a quantidade de sessões ativas (sem uma data e hora de término) criadas por este jogador.*

```

CREATE OR REPLACE TRIGGER ASSURE_NO_UNFINISHED_SESSION BEFORE
    INSERT ON GAME_SESSION FOR EACH ROW
DECLARE
    SESSION_COUNT INT;
BEGIN
    SELECT
        COUNT_ACTIVE_SESSIONS(:NEW.ID_USER) INTO SESSION_COUNT
    FROM
        DUAL;
    IF SESSION_COUNT >= 1 THEN
        RAISE_APPLICATION_ERROR (-20101, 'Game_session: A player can''t have more than one active session at a time.');
    END IF;
END;
/

```

*Criação do trigger Assure No Unfinished Session. É responsável por assegurar que há apenas uma única sessão não terminada por jogador. Coloca um erro caso tenha.*

```
CREATE OR REPLACE FUNCTION GET_SINGLEPLAYER_CODE (
    CODE_GAME INT
) RETURN INT AS
    CODE_SP_GAME INT;
BEGIN
    SELECT
        CODE_GAME INTO CODE_SP_GAME
    FROM
        SINGLEPLAYER_GAME
    WHERE
        CODE_GAME = GET_SINGLEPLAYER_CODE.CODE_GAME;
    RETURN CODE_SP_GAME;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/
```

*Criação da função Get Singleplayer Code. Recebe o código de um jogo como parâmetro e devolve o próprio código caso este seja um jogo do tipo singleplayer, caso contrário, retorna NULL.*

```

CREATE OR REPLACE TRIGGER ASSURE_GAME_IS_VALID BEFORE
| INSERT ON GAME FOR EACH ROW
DECLARE
    CODE_SP_GAME INT;
    CODE_MP_GAME INT;
BEGIN
    SELECT
        | GET_SINGLEPLAYER_CODE(:NEW.CODE_GAME) INTO CODE_SP_GAME
    FROM
        | DUAL;
    SELECT
        | GET_MULTIPLAYER_CODE(:NEW.CODE_GAME) INTO CODE_MP_GAME
    FROM
        | DUAL;
    IF CODE_SP_GAME IS NULL AND CODE_MP_GAME IS NULL THEN
        | RAISE_APPLICATION_ERROR (-20101, 'Game: A game must be single-player, multi-player or both.');
    END IF;
END;
/

```

*Criação do trigger Assure Game Is Valid.* É responsável pela verificação se um jogo é exclusivamente multiplayer ou singleplayer na criação de um jogo (inserção na tabela GAME). Coloca um erro caso não se valide.

```

CREATE OR REPLACE TRIGGER ASSURE_NO_UNFINISHED_SESSION BEFORE
| INSERT ON GAME_SESSION FOR EACH ROW
DECLARE
    SESSION_COUNT INT;
BEGIN
    SELECT
        | COUNT_ACTIVE_SESSIONS(:NEW.ID_USER) INTO SESSION_COUNT
    FROM
        | DUAL;
    IF SESSION_COUNT >= 1 THEN
        | RAISE_APPLICATION_ERROR (-20101, 'Game_session: A player can''t have more than one active session at a time.');
    END IF;
END;
/

```

*Criação do trigger Assure No Unfinished Session.* Quando uma sessão é criada, verifica se o usuário que criou a sessão não tem uma outra sessão não terminada. Coloca um erro caso tenha.

```

CREATE OR REPLACE PROCEDURE ADD_STUDIO (
    NIF_COMPANY VARCHAR2,
    NAME_COMPANY VARCHAR2,
    DESCRIPTION_COMPANY VARCHAR2,
    COMPANY_ADDRESS VARCHAR2,
    NIF_PARENT VARCHAR2
) AS
BEGIN
    INSERT INTO COMPANY VALUES(
        NIF_COMPANY,
        NAME_COMPANY,
        DESCRIPTION_COMPANY,
        COMPANY_ADDRESS
    );
    INSERT INTO STUDIO VALUES(
        NIF_COMPANY
    );
    IF NIF_PARENT IS NOT NULL THEN
        ADD_SUBSIDIARY(NIF_COMPANY, NIF_PARENT);
    END IF;
END;
/

```

*Criação do procedimento Add Studio. Recebe como parâmetros o NIF, nome, descrição, endereço e opcionalmente o NIF da empresa parente. Cria uma compania com os dados recebidos e classifica-a como estúdio.*

```

CREATE OR REPLACE PROCEDURE ADD_PUBLISHER (
    NIF_COMPANY VARCHAR2,
    NAME_COMPANY VARCHAR2,
    DESCRIPTION_COMPANY VARCHAR2,
    COMPANY_ADDRESS VARCHAR2,
    NIF_PARENT VARCHAR2
) AS
BEGIN
    INSERT INTO COMPANY VALUES(
        NIF_COMPANY,
        NAME_COMPANY,
        DESCRIPTION_COMPANY,
        COMPANY_ADDRESS
    );
    INSERT INTO PUBLISHER VALUES(
        NIF_COMPANY
    );
    IF NIF_PARENT IS NOT NULL THEN
        ADD_SUBSIDIARY(NIF_COMPANY, NIF_PARENT);
    END IF;
END;
/

```

*Criação do procedimento Add Publisher. Recebe como parâmetros o NIF, nome, descrição e opcionalmente o NIF da empresa parente. Cria uma compania com os dados recebidos e classifica-a como publicadora.*

```

CREATE OR REPLACE PROCEDURE CREATE_USER (
    ID_USER VARCHAR2,
    NAME_USER VARCHAR2,
    EMAIL VARCHAR2,
    USER_PASSWORD VARCHAR2,
    USER_ADDRESS VARCHAR2,
    NIF_DEVELOPER VARCHAR2,
    DATE_OF_BIRTH TIMESTAMP,
    PAYMENT_METHOD VARCHAR2
) AS
BEGIN
    IF PAYMENT_METHOD IS NULL AND NIF_DEVELOPER IS NULL THEN
        RAISE_APPLICATION_ERROR (-20101, 'Create_user: Must have a nif number and/or a payment method');
    END IF;
    INSERT INTO GAME_USER VALUES(
        ID_USER,
        NAME_USER,
        EMAIL,
        USER_PASSWORD,
        DATE_OF_BIRTH,
        USER_ADDRESS
    );
    IF NIF_DEVELOPER IS NOT NULL THEN
        INSERT INTO DEVELOPER VALUES(
            ID_USER,
            NIF_DEVELOPER
        );
    END IF;
    IF PAYMENT_METHOD IS NOT NULL THEN
        INSERT INTO PLAYER VALUES(
            ID_USER,
            PAYMENT_METHOD
        );
    END IF;
END;
/

```

*Criação do procedimento Create User. Recebe como parâmetros o ID, nome, email, senha, endereço, NIF de desenvolvedor caso seja um, data de nascimento e método de pagamento caso seja um jogador. Cria um usuário novo com os dados fornecidos e regista-os como o tipo apropriado, jogador e/ou desenvolvedor.*

```
CREATE OR REPLACE PROCEDURE UPDATE_USER_EMAIL (
    ID_USER VARCHAR2,
    EMAIL VARCHAR2
) AS
BEGIN
    UPDATE GAME_USER
    SET
        EMAIL = UPDATE_USER_EMAIL.EMAIL
    WHERE
        ID_USER = UPDATE_USER_EMAIL.ID_USER;
END;
/
```

*Criação do procedimento Update User Email. Recebe como parâmetros o ID do usuário e o email novo. Atualiza o email do usuário indicado para o fornecido.*

```
CREATE OR REPLACE PROCEDURE UPDATE_USER_NAME (
    ID_USER VARCHAR2,
    NAME_USER VARCHAR2
) AS
BEGIN
    UPDATE GAME_USER
    SET
        NAME_USER = UPDATE_USER_NAME.NAME_USER
    WHERE
        ID_USER = UPDATE_USER_NAME.ID_USER;
END;
/
```

*Criação do procedimento Update User Name. Recebe como parâmetros o ID do usuário e o novo nome. Atualiza o nome do usuário indicado para o fornecido.*

```
CREATE OR REPLACE PROCEDURE UPDATE_USER_PASSWORD (
    ID_USER VARCHAR2,
    USER_PASSWORD VARCHAR2
) AS
BEGIN
    UPDATE GAME_USER
    SET
        USER_PASSWORD = UPDATE_USER_PASSWORD.USER_PASSWORD
    WHERE
        ID_USER = UPDATE_USER_PASSWORD.ID_USER;
END;
/
```

*Criação do procedimento Update User Email. Recebe como parâmetros o ID do usuário e a senha nova. Atualiza a senha do usuário indicado para o fornecido.*

```
CREATE OR REPLACE PROCEDURE UPDATE_USER_ADDRESS (
    ID_USER VARCHAR2,
    USER_ADDRESS VARCHAR2
) AS
BEGIN
    UPDATE GAME_USER
    SET
        USER_ADDRESS = UPDATE_USER_ADDRESS.USER_ADDRESS
    WHERE
        ID_USER = UPDATE_USER_ADDRESS.ID_USER;
END;
/
```

*Criação do procedimento Update User Email. Recebe como parâmetros o ID do usuário e o endereço novo. Atualiza o endereço do usuário indicado para o fornecido.*

```
CREATE OR REPLACE FUNCTION IS_PLAYER (
    ID_USER VARCHAR2
) RETURN VARCHAR2 AS
    PLAYER_ID VARCHAR2(30);
BEGIN
    SELECT
        P.ID_USER INTO PLAYER_ID
    FROM
        PLAYER P
    WHERE
        P.ID_USER = IS_PLAYER.ID_USER;
    RETURN PLAYER_ID;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/
```

*Criação da função Is Player. Recebe como parâmetro o ID de um usuário e retorna este mesmo ID se este for do tipo jogador, NULL caso contrário.*

```

CREATE OR REPLACE PROCEDURE UPDATE_PAYMENT_METHOD (
    ID_USER VARCHAR2,
    PAYMENT_METHOD VARCHAR2
) AS
BEGIN
    IF IS_PLAYER(ID_USER) IS NULL THEN
        INSERT INTO PLAYER VALUES(
            ID_USER,
            PAYMENT_METHOD
        );
    ELSE
        UPDATE PLAYER
        SET
            PAYMENT_METHOD = UPDATE_PAYMENT_METHOD.PAYMENT_METHOD
        WHERE
            ID_USER = UPDATE_PAYMENT_METHOD.ID_USER;
    END IF;
END;
/

```

*Criação do procedimento Update Payment Method. Recebe como parâmetros o ID do usuário e o novo método de pagamento. Atualiza o método de pagamento do usuário fornecido.*

```
CREATE OR REPLACE FUNCTION IS_DEVELOPER (
    ID_USER VARCHAR2
) RETURN VARCHAR2 AS
    DEVELOPER_ID VARCHAR2(30);
BEGIN
    SELECT
        D.ID_USER INTO DEVELOPER_ID
    FROM
        DEVELOPER D
    WHERE
        D.ID_USER = IS_DEVELOPER.ID_USER;
    RETURN DEVELOPER_ID;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/
```

*Criação da função Is Developer.*

*Espelho da função Is Player, recebe como parâmetro o ID do usuário e retorna o mesmo ID se este for do tipo desenvolvedor, NULL caso contrário.*

```

CREATE OR REPLACE PROCEDURE UPDATE_NIF_DEVELOPER (
    ID_USER VARCHAR2,
    NIF_DEVELOPER VARCHAR2
) AS
BEGIN
    IF IS_DEVELOPER(ID_USER) IS NULL THEN
        INSERT INTO DEVELOPER VALUES(
            ID_USER,
            NIF_DEVELOPER
        );
    ELSE
        UPDATE DEVELOPER
        SET
            NIF_DEVELOPER = UPDATE_NIF_DEVELOPER.NIF_DEVELOPER
        WHERE
            ID_USER = UPDATE_NIF_DEVELOPER.ID_USER;
    END IF;
END;
/

```

*Criação do procedimento Update Nif Developer. Recebe como parâmetros o ID do usuário e o NIF de desenvolvedor a atualizar. Verifica se o usuário já é do tipo desenvolvedor, caso seja seu NIF é atualizado. Caso contrário, é registado como desenvolvedor e seu NIF é definido.*

```

CREATE OR REPLACE FUNCTION HAS_EMPLOYER (
    ID_USER VARCHAR2
) RETURN VARCHAR2 AS
    EMPLOYER_NIF VARCHAR2(9);
BEGIN
    IF IS_DEVELOPER(ID_USER) IS NULL THEN
        RETURN NULL;
    END IF;
    SELECT
        W.NIF_COMPANY INTO EMPLOYER_NIF
    FROM
        WORKS W
    WHERE
        W.ID_USER = HAS_EMPLOYER.ID_USER;
    RETURN EMPLOYER_NIF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
/

```

*Criação da função Has Employer. Recebe como parâmetro o ID do usuário. Retorna o NIF de desenvolvedor se o usuário tiver um estúdio como empregador (estar na tabela Works), NULL caso o usuário não seja um desenvolvedor ou não esteja empregado em nenhum estúdio.*

```

CREATE OR REPLACE PROCEDURE UPDATE_EMPLOYER (
    ID_USER VARCHAR2,
    NIF_STUDIO VARCHAR2
) AS
BEGIN
    IF HAS_EMPLOYER(ID_USER) IS NULL THEN
        INSERT INTO WORKS VALUES (
            ID_USER,
            NIF_STUDIO
        );
    ELSE
        UPDATE WORKS
        SET
            NIF_COMPANY = UPDATE_EMPLOYER.NIF_STUDIO
        WHERE
            ID_USER = UPDATE_EMPLOYER.ID_USER;
    END IF;
END;
/

```

*Criação do procedimento Update Employer. Recebe como parâmetros o ID do usuário e o NIF do estúdio a trabalhar. Se o desenvolvedor não estiver trabalhando em nenhum estúdio, é criado um tuplo na tabela Works para ele. Caso contrário, apenas atualiza o estúdio em que trabalha.*

Quando adicionamos um jogo na plataforma, devemos avaliar diversas tabelas e atualizá-las. Dessa forma, tratamos a inserção de jogos utilizando os seguintes procedimentos e tipos auxiliares:

```
CREATE OR REPLACE TYPE SELECTED_GENRES AS
  VARRAY(
    16
  ) OF INT;
/

CREATE OR REPLACE TYPE SELECTED_PARTNERS AS
  VARRAY(
    10
  ) OF VARCHAR2(
    9
);
/

CREATE OR REPLACE TYPE SELECTED_DEVELOPERS AS
  VARRAY(
    10
  ) OF VARCHAR2(
    30
);
/
```

*Definição dos arrays que armazenam os gêneros, estúdios parceiros e desenvolvedores (caso o não tenha estúdio).*

```
CREATE OR REPLACE PROCEDURE INSERT_GAME (
    CODE_GAME INT,
    NAME_GAME VARCHAR2,
    DESC_GAME VARCHAR2,
    RELEASE_DATE DATE,
    CURRENT_PRICE DECIMAL
) AS
BEGIN
    INSERT INTO GAME VALUES (
        CODE_GAME,
        NAME_GAME,
        DESC_GAME,
        RELEASE_DATE,
        CURRENT_PRICE
    );
END;
/
```

*Criação do procedimento Insert Game. Recebe como parâmetro o código, nome, descrição, data de lançamento e preço. Cria um novo jogo na tabela Game com os dados recebidos.*

```

CREATE OR REPLACE PROCEDURE INSERT_GENRES (
    CODE_GAME INT,
    CODE_GENRE IN SELECTED_GENRES
) AS
BEGIN
    FOR I IN 1..CODE_GENRE.COUNT LOOP
        INSERT INTO HAS_GENRE VALUES (
            CODE_GAME,
            CODE_GENRE(I)
        );
    END LOOP;
END;
/

```

*Criação do procedimento Insert Genres. Recebe como parâmetro o código do jogo e um array de códigos de gêneros. Regista os gêneros correspondentes aos códigos recebidos no jogo indicado (inserção na tabela Has Genre).*

```

CREATE OR REPLACE PROCEDURE INSERT_DEV_CREATES (
    CODE_GAME INT,
    ID_DEVELOPER IN SELECTED_DEVELOPERS
) AS
BEGIN
    FOR I IN 1..ID_DEVELOPER.COUNT LOOP
        INSERT INTO DEV_CREATES VALUES (
            CODE_GAME,
            ID_DEVELOPER(I)
        );
    END LOOP;
END;
/

```

*Criação do procedimento Insert Dev Creates. Recebe como parâmetros o código do jogo e um array de IDs de desenvolvedores. Regista todos os IDs enviados como desenvolvedores do jogo indicado.*

```

CREATE OR REPLACE PROCEDURE ADD_GAME (
    NAME_GAME VARCHAR2,
    DESC_GAME VARCHAR2,
    RELEASE_DATE DATE,
    CODE_GENRE IN SELECTED_GENRES,
    NIF_STUDIO VARCHAR2, -- case null if isn't created by a studio
    NIF_PARTNERS IN SELECTED_PARTNERS, -- case null if don't have partners
    ID_DEVELOPER IN SELECTED_DEVELOPERS, -- case null if isn't created by a team of developers
    NIF_PUBLISHER VARCHAR2,
    CURRENT_PRICE DECIMAL,
    IS_SINGLEPLAYER BOOLEAN,
    IS_MULTIPLAYER BOOLEAN
) AS
    CODE_GAME INT; -- enumerates the game entries
BEGIN
    IF NOT IS_SINGLEPLAYER AND NOT IS_MULTIPLAYER THEN
        RAISE_APPLICATION_ERROR(-20101, 'Add_game: Game must be single_player, multiplayer or both');
    END IF;
    EXECUTE IMMEDIATE 'ALTER SESSION SET CONSTRAINTS = DEFERRED';
    IF CODE_GENRE IS NULL OR CODE_GENRE.COUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20101, 'Add_game: Must have a genre.');
    END IF;
    SELECT
        COALESCE(MAX(CODE_GAME),
        0) + 1 INTO CODE_GAME
    FROM
        GAME;
    INSERT_GENRES(CODE_GAME, CODE_GENRE);
    IF NIF_STUDIO IS NOT NULL THEN
        INSERT_CREATES(CODE_GAME, NIF_STUDIO, NIF_PARTNERS, NIF_PUBLISHER);
    ELSE
        IF ID_DEVELOPER IS NULL OR ID_DEVELOPER.COUNT = 0 THEN
            RAISE_APPLICATION_ERROR(-20102, 'Add_game: If game is not made by a studio, it must have developers.');
        END IF;
        INSERT_DEV_CREATES(CODE_GAME, ID_DEVELOPER);
    END IF;
    IF IS_SINGLEPLAYER THEN
        INSERT INTO SINGLEPLAYER_GAME VALUES(
            CODE_GAME
        );
    END IF;
    IF IS_MULTIPLAYER THEN
        INSERT INTO MULTIPLAYER_GAME VALUES(
            CODE_GAME
        );
    END IF;
    INSERT_GAME(CODE_GAME, NAME_GAME, DESC_GAME, RELEASE_DATE, CURRENT_PRICE);
    COMMIT;
    EXECUTE IMMEDIATE 'ALTER SESSION SET CONSTRAINTS = IMMEDIATE';
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        EXECUTE IMMEDIATE 'ALTER SESSION SET CONSTRAINTS = IMMEDIATE';
        RAISE;
        COMMIT;
END;
/

```

*Descrição na página seguinte*

*Criação do procedimento Add Game. Recebe todas as informações relacionadas a um jogo: nome, descrição, data de lançamento, um array de códigos de gêneros, o NIF do estúdio que o criou (quando aplicável), O NIF dos estúdios parceiros na sua criação (quando aplicável), os IDs dos desenvolvedores independentes (quando aplicável), o NIF da publicadora (quando aplicável), o seu preço atual, se ele é singleplayer ou não, se ele é multiplayer ou não. Insere um novo jogo com os dados recebidos na base de dados. Além disso, regista todas as suas informações relevantes em outras tabelas: os seus gêneros (as\_genre), os estúdios principais (creates) e parceiros (has\_partners) ou os desenvolvedores independentes (devCreates) responsáveis pela sua criação e finalmente regista-o como um jogo singleplayer (singleplayer\_game) e/ou multiplayer (multiplayer\_game). Coloca um erro quando não há nenhuma definição de singleplayer/multiplayer, não há um gênero, não há criadores associados (nem estúdios nem desenvolvedores independentes).*

*Por último, altera a sessão e coloca as constraints como IMMEDIATE, aplicando assim as restrições nos atributos anteriormente definidos como DEFERRABLE.*

*É importante ressaltar que não há possibilidade de inserir um jogo diretamente através da aplicação. Esta função é usada para inserir exemplos de jogos utilizando SQL.*

```
CREATE OR REPLACE FUNCTION GET_PRICE (
    CODE_GAME INT
) RETURN NUMBER AS
    PRICE DECIMAL(10, 2);
BEGIN
    SELECT
        CURRENT_PRICE INTO PRICE
    FROM
        GAME
    WHERE
        CODE_GAME = GET_PRICE.CODE_GAME;
    RETURN PRICE;
END;
/
```

*Criação da função Get Price. Recebe como parâmetro o código do jogo e retorna o preço deste jogo.*

```

CREATE OR REPLACE FUNCTION GET_PAYMENT_METHOD (
    ID_USER VARCHAR2
) RETURN VARCHAR2 AS
    PAYMENT_METHOD VARCHAR2(200);
BEGIN
    SELECT
        PAYMENT_METHOD INTO PAYMENT_METHOD
    FROM
        PLAYER
    WHERE
        ID_USER = GET_PAYMENT_METHOD.ID_USER;
    RETURN PAYMENT_METHOD;
END;
/

```

*Criação da função Get Payment Method. Recebe o ID do usuário e retorna o seu método de pagamento atual.*

```

CREATE OR REPLACE PROCEDURE PURCHASE_GAME (
    CODE_GAME INT,
    ID_USER VARCHAR2,
    PURCHASE_DATE TIMESTAMP
) AS
BEGIN
    INSERT INTO OWNS VALUES (
        CODE_GAME,
        ID_USER,
        GET_PRICE(CODE_GAME),
        PURCHASE_DATE,
        GET_PAYMENT_METHOD(ID_USER)
    );
END;
/

```

*Criação do procedimento Purchase Game. Recebe como parâmetros o código do jogo, o ID do usuário e a data de compra. Regista o jogo indicado como comprado pelo usuário referido pelo ID, na data enviada (inserção na tabela Owns).*

```
CREATE OR REPLACE FUNCTION GENERATE_SERVER_ID (
    CODE_GAME INT
) RETURN INT AS
    SERVER_ID INT;
BEGIN
    SELECT
        COUNT(*) + 1 INTO SERVER_ID
    FROM
        GAME_SERVER
    WHERE
        CODE_GAME = GENERATE_SERVER_ID.CODE_GAME;
    RETURN SERVER_ID;
END;
/
```

Criação da função Generate Server ID. Recebe como parâmetro o código do jogo. Retorna um ID de servidor (associado à tabela game\_server) para o jogo indicado. Os IDs de servidores são sequenciais, o primeiro servidor tem ID 1, o segundo 2, etc. Por causa disto, o ID a ser devolvido é a quantidade de servidores associados à aquele jogo incrementada.

```

CREATE OR REPLACE PROCEDURE ADD_SERVER (
    CODE_GAME INT,
    DATETIME_CREATION TIMESTAMP,
    REGION VARCHAR2,
    ID_USER VARCHAR2, -- Can be null when the server is not created by a player
    HOST_KEY VARCHAR2 -- Can be null when the server is not created by a player
) AS
    SERVER_ID INT;
BEGIN
    IF GET_MULTIPLAYER_CODE(CODE_GAME) IS NULL THEN
        RAISE_APPLICATION_ERROR(-20101, 'ADD_SERVER: Game must be a multi-player game.');
    END IF;
    SELECT
        GENERATE_SERVER_ID(CODE_GAME) INTO SERVER_ID
    FROM
        DUAL;
    INSERT INTO GAME_SERVER VALUES(
        CODE_GAME,
        SERVER_ID,
        DATETIME_CREATION,
        REGION
    );
    IF ID_USER IS NOT NULL THEN
        IF HOST_KEY IS NULL THEN
            RAISE_APPLICATION_ERROR(-20101, 'ADD_SERVER: Player created servers must have a non-null host_key.');
        END IF;
        INSERT INTO CREATES_SERVER VALUES(
            CODE_GAME,
            SERVER_ID,
            ID_USER,
            HOST_KEY
        );
    END IF;
END;
/

```

*Criação do procedimento Create Server. Recebe como parâmetros o código do jogo, a data e hora de criação, a região, o ID de usuário do criador e a chave de hospedagem (quando o servidor é criado por um player). Cria um servidor com os dados fornecidos, se este não for criado por um player (ID de usuário NULL), não lhe é associado nenhum player (não é inserido na tabela creates\_server). Coloca um erro se o jogo não for do tipo multiplayer ou se o servidor for criado por um player (ID de usuário diferente de NULL) não tiver uma chave de hospedagem definida (host\_key NULL).*

```

CREATE OR REPLACE PROCEDURE ADD_SESSION (
    ID_USER VARCHAR2,
    CODE_GAME NUMBER,
    DATE_START TIMESTAMP,
    SERVER_ID NUMBER -- Null when the session is single-player
) AS
BEGIN
    IF SERVER_ID IS NOT NULL AND GET_MULTIPLAYER_CODE(CODE_GAME) IS NULL THEN
        RAISE_APPLICATION_ERROR(-20101, 'ADD_SESSION: Games that are only single-player don''t have multiplayer sessions.');
    END IF;
    INSERT INTO GAME_SESSION VALUES(
        ID_USER,
        CODE_GAME,
        DATE_START,
        NULL
    );
    IF SERVER_ID IS NOT NULL THEN
        INSERT INTO MULTIPLAYER_SESSION VALUES(
            ID_USER,
            CODE_GAME,
            DATE_START,
            SERVER_ID,
            CODE_GAME
        );
    ELSE
        INSERT INTO SINGLEPLAYER_SESSION VALUES(
            ID_USER,
            CODE_GAME,
            DATE_START
        );
    END IF;
    COMMIT;
END;
/

```

*Criação do procedimento Add Session. Recebe como parâmetros o ID de usuário, o código do jogo, a data e hora de início e o ID do servidor(caso seja multiplayer). Cria uma sessão com os dados fornecidos. A regista como sessão singleplayer se não houver um ID de servidor. Caso contrário, a regista como uma sessão multiplayer. Coloca um erro caso seja enviada um ID de servidor de um jogo exclusivamente singleplayer.*

```

CREATE OR REPLACE PROCEDURE END_SESSION (
    ID_USER VARCHAR2
) AS
BEGIN
    UPDATE GAME_SESSION
    SET
        DATETIME_END = SYSTIMESTAMP
    WHERE
        ID_USER = END_SESSION.ID_USER
        AND DATETIME_END IS NULL;
    COMMIT;
END;
/

```

*Criação do procedimento End Session. Recebe como parâmetro o ID do usuário e termina a sessão concorrente deste usuário. Como cada usuário só pode ter uma sessão interminada, não é necessário procurar a sessão correta.*

## Inserções e Exemplos

Como visto acima, alguns dos dados da plataforma são inseridos através de procedimentos, portanto algumas das inserções já foram demonstradas acima.

A seguir, temos como são realizadas as inserções naturais de algumas tabelas e como funciona a chamada daquelas que são os procedimentos, exemplos:

```
----- ##  
----  
-- Insertion of COMPANIES -- not necessary publishers or studios  
-- Args: nif, name, desc, addr  
INSERT INTO COMPANY VALUES (  
    '549656558',  
    'Microsoft',  
    'The largest vendor of computers in the world, owner of Xbox ',  
    'Washington - USA'  
);  
  
INSERT INTO COMPANY VALUES (  
    '557375112',  
    'Sony',  
    'The leader in audio/video electronics, owner of Playstation',  
    'Tokyo - JAPAN'  
);
```

*Criação de exemplos de empresas.*

```
----  
-- Insertion of PUBLISHER -- must follow the pattern  
-- Args: nif , name , desc , addr , nif_parent (specify if has one, NULL otherwise)  
ADD_PUBLISHER('466666666', 'Rockstar Games', 'To be announced', 'New York - USA', '531093041');  
ADD_PUBLISHER('467949834', 'BANDAI NAMCO', 'We make unique games', 'Tokyo - Japan', NULL);  
ADD_PUBLISHER('467949835', 'Valve', 'One, two but never three', 'Washington - USA', '566997193');
```

*Criação de exemplos de empresas publicadoras.*

```
-----
-- Insertion of STUDIOS -- must follow the pattern
-- Args: nif , name , desc , addr , nif_parent (specify if has one, NULL otherwise)
ADD_STUDIO('366666552', 'Rockstar North', 'Working on that game', 'Edinburgh, Scotland', '466666666');
ADD_STUDIO('366666217', 'Rockstar India', 'Helping that game', 'Bangalore - India', '466666666');
ADD_STUDIO('336666216', 'Rockstar Toronto', 'Not doing that game', 'Toronto - Canada', '466666666');
```

*Criação de exemplos de estúdios.*

```
-----
-- Insertion of USERS -- must specify if is player, dev or both
-- Args: id , name , email , password , address, nif (case is a dev, null otherwise), payment method (case is a player, null otherwise)
CREATE_USER('DoeJohn', 'John Doe', 'john.doe@gmail.com', 'password1bad', 'Texas - USA', '123456789', TO_TIMESTAMP('1990-01-01', 'YYYY-MM-DD'), 'Credit');
CREATE_USER('JanePlays', 'Jane Smith', 'jane.smith@hotmail.com', 'catfurry78', 'Galway - Ireland', '987654321', TO_TIMESTAMP('1995-05-15', 'YYYY-MM-DD'), NULL);
CREATE_USER('MikeRules', 'Michael Johnson', 'michael.johnson@gmail.com', 'johnsonJ', 'Coimbra - Portugal', '234567890', TO_TIMESTAMP('1988-10-20', 'YYYY-MM-DD'), 'Debit');
CREATE_USER('SarahCRW', 'Sarah Williams', 'sarah.williams@outlook.com', 'll39015s', 'Aveiro - Portugal', '876543210', TO_TIMESTAMP('1992-06-30', 'YYYY-MM-DD'), 'PayPal');
```

*Criação de exemplos de usuários.*

```
-----
-- Insertion of WORKS -- associates a dev with a studio
-- Args: developer id, nif company
INSERT INTO WORKS VALUES (
    'GOAT',
    '377807780'
);

INSERT INTO WORKS VALUES (
    'Amorinus',
    '391797921'
);
```

*Criação de exemplos de registos de trabalho de desenvolvedores em um estúdio.*

```
-----
-- Insertion of GENRES --
-- Args: code, name
INSERT INTO GENRE VALUES (
    1,
    'Action'
);

INSERT INTO GENRE VALUES (
    2,
    'Adventure'
);
```

*Criação de exemplos de gêneros.*

```
-- Insertion of GAMES -- many possible cases
-- Args: game title, desc, release date, genres, nif studio, nif partners, nif developers, nif publisher, price, if is singleplayer, if is multiplayer
-- Use SELECTED_GENRES() when informing the genres
-- Use SELECTED_PARTNERS() when informing studios that helps the development
-- Use SELECTED_DEVELOPERS() when informing the developers
ADD_GAME('Foamstars', 'A copy of Splatoon', TO_TIMESTAMP('2016-06-15', 'YYYY-MM-DD'), SELECTED_GENRES(1, 8), '318721904', NULL, NULL, NULL, 59.99, FALSE, TRUE);
ADD_GAME('Minecraft 2', 'Much more pixels and a new way of seeing things, experience surviving in a new blocky world', TO_TIMESTAMP('2016-09-20', 'YYYY-MM-DD'), SELECTED_GENRES(1, 2, 15), '365270912', NULL, NULL, '464137789', 39.99, TRUE, TRUE);
ADD_GAME('Stardew Valley 2', 'Now you can make that thing', TO_TIMESTAMP('2017-07-25', 'YYYY-MM-DD'), SELECTED_GENRES(2, 5), NULL, NULL, SELECTED_DEVELOPERS('BaronH'), NULL, 24.99, TRUE, TRUE);
ADD_GAME('The Elder Scrolls VI', 'Embark on a new epic adventure in the world of Tamriel', TO_TIMESTAMP('2018-11-11', 'YYYY-MM-DD'), SELECTED_GENRES(1, 2, 5), '391797921', NULL, NULL, '439740932', 49.99, TRUE, FALSE);
ADD_GAME('The Witcher 3: Wild Hunt', 'Embark on a dark and immersive journey as Geralt of Rivia in this epic RPG', TO_TIMESTAMP('2015-05-19', 'YYYY-MM-DD'), SELECTED_GENRES(1, 2, 5), '375929402', NULL, NULL, NULL, 29.99, TRUE, FALSE);
```

*Criação de exemplos de jogos.*

```
-----
-- Insertion of OWNS
-- Args: game code, user id, purchase price, purchase date
PURCHASE_GAME(3, 'GOAT', SYSDATE);
PURCHASE_GAME(13, 'DoeJohn', SYSDATE);
PURCHASE_GAME(13, 'Inbox', SYSDATE);
```

*Criação de exemplos de registos de compras.*

```
-----
-- Insertion of REVIEWS
-- Args: user id, game code, review date, rating, review text
INSERT INTO REVIEWS VALUES(
    'GOAT',
    3,
    CURRENT_TIMESTAMP,
    4,
    'WHY ARE THERE SO MANY PLANTS'
);

INSERT INTO REVIEWS VALUES(
    'DoeJohn',
    13,
    CURRENT_TIMESTAMP,
    5,
    'Played this game so much I''m clipping through the walls irl, 10/10'
);
```

*Criação de exemplos de reviews.*

```
-----
-- Insertion of SERVERS --
-- Args: game code, date, region, id user, host key
ADD_SERVER(2, SYSDATE, 'US', NULL, NULL);
ADD_SERVER(2, SYSDATE, 'EU', 'Inbox', 'ABC123');
ADD_SERVER(18, SYSDATE, 'EU', 'TechNinja', 'DEF456');
```

*Criação de exemplos de servidores.*

```
-----
-- Insertion of SESSION -- to end a session use END_SESSION with the id user
-- Args: id user, code game, init date, server id (if has one, null otherwise)
ADD_SESSION('GOAT', 3, SYSDATE, NULL);
END_SESSION('GOAT');
```

*Criação de um exemplo de sessão terminada.*

```

-----
-- Update User Info -- calls the procedure that corresponds to what you want to update
-- Args: User ID, the information to update (Address, email, name, password, nif, payment method)
UPDATE_USER_ADDRESS('SarahCRW', 'Singapore - Singapore');
UPDATE_USER_EMAIL('PixelArtist', 'm.thompson@eyefind.com');
UPDATE_USER_NAME('JanePlays', 'Jane Doe');
UPDATE_USER_PASSWORD('Plant', 'ImDeFiNeTiLyNoTMaKingHL3');
UPDATE_NIF_DEVELOPER('EmilyProfD', '777320195');
UPDATE_PAYMENT_METHOD('MikeRules', 'MBWay');

```

*Criação de exemplos de atualizações de dados de usuários.*

## Opções tomadas na implementação da Plataforma

Para garantir a integridade da base de dados tivemos de implementar vários triggers. Por exemplo, para garantir que um jogo desenvolvido por um estúdio não tivesse como parceiro aquele mesmo estúdio criamos o trigger IS\_NOT\_MAIN\_STUDIO. Procuramos evitar criar demasiados triggers quando não fosse estritamente necessário para que a performance da Base de Dados não sofresse, por exemplo, em situações que utilizamos o CHECK na criação de tabelas.

Certas entradas em tabelas requerem entradas previamente existentes em outras tabelas, como no caso dos jogos (que requerem a presença de pelo menos uma entrada com o código do jogo e um código de género em Has\_Genre). Como anteriormente mencionamos, triggers e constraints que garantem a integridade da base de dados tornam o processo de criação do jogo mais complexo. Para agilizar este processo, tornamos as constraints que dependem do jogo deferrable nas tabelas que são modificadas na criação do jogo, e criamos um procedure (*Add\_Game*) que insere automaticamente toda a informação necessária nas tabelas adequadas. Procedimentos semelhantes foram criados para criar estúdios (*Add\_Studio*), distribuidoras (*Add\_Publisher*), utilizadores (*Add\_user*), entre outros.

Para não complicar demasiadamente a base de dados, não entramos em detalhes sobre a efetuação de pagamentos feitos pelos jogadores na compra de jogos, assim como não detalhamos os métodos de pagamento (que estão presentes apenas como valores de texto).

# Consultas interessantes

Algumas dessas consultas estão distribuídas por toda a plataforma e serão explicitadas novamente quando for explicado na interface APEX.

- **Lista dos 5 jogos com melhor avaliação**

The Best  
These are our 5 best rated games of all time.

Game	Rating
Halo Infinite	5
Cybernetic Arena	5
Minor Pay Manual	4
Resident Evil 9	4
Stardew Valley 2	3.5

## *Top 5 jogos com melhor avaliação*

Na página What's new ([página 6 - Stats](#)) quisemos mostrar os 5 jogos mais bem avaliados e a sua respetiva média de avaliação.

```
SELECT * FROM (SELECT
    NAME_GAME AS GAME,
    AVG(RATING) AS RATING
FROM
    GAME
    NATURAL JOIN OWNS NATURAL JOIN REVIEWS
GROUP BY
    NAME_GAME
ORDER BY RATING DESC) WHERE ROWNUM < 6;
```

*Código correspondente*

- **Estúdios envolvidos num jogo**

Developers and Publishers

Studio(s): Rockstar India:Rockstar North:Rockstar Toronto

Publisher: Rockstar Games

*Exemplo de jogo desenvolvido com a participação de estúdios parceiros*

Queremos mostrar de forma simples todos os estúdios que participaram da criação de um jogo na página de jogos (**página 2 - Game Page**). É necessário, portanto, mostrar o estúdio que está na relação *creates* juntamente aos estúdios que estão na relação *has\_partners*.

```
SELECT
    DISTINCT CP.NAME_COMPANY
FROM
    STUDIO S INNER JOIN COMPANY CP ON S.NIF_COMPANY = CP.NIF_COMPANY
    INNER JOIN CREATES C
    ON S.NIF_COMPANY = C.NIF_STUDIO
WHERE
    C.CODE_GAME = :P2_CODE
UNION
SELECT
    DISTINCT CP.NAME_COMPANY
FROM
    STUDIO S INNER JOIN COMPANY CP ON S.NIF_COMPANY = CP.NIF_COMPANY
    INNER JOIN HAS_PARTNERS HP
    ON S.NIF_COMPANY = HP.NIF_COMPANY
WHERE
    HP.CODE_GAME = :P2_CODE;
```

#### *Código correspondente*

É necessário notar que o código do jogo (*code\_game*) está localizado na variável *:P2\_CODE*.

- **Lista dos 5 jogos mais vendidos**

Hall of fame	
These are our 5 best selling games of all time.	
Game	Sales
Cyberpunk 2077	3
Halo Infinite	2
Stardew Valley 2	2
Life Stranding	1
Call of Duty: Black Ops II Remastered	1

#### *Lista dos 5 jogos mais vendidos*

Na página What's new (**página 6 - Stats**) quisemos mostrar os 5 jogos mais vendidos, assim como o seu número de vendas.

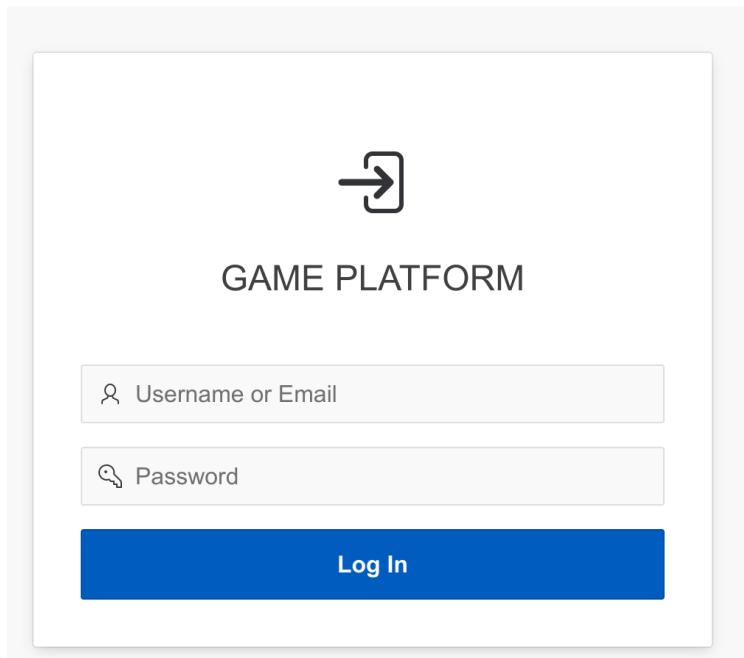
```
SELECT * FROM (SELECT
    NAME_GAME AS GAME,
    COUNT(ID_USER) AS SALES
  FROM
    GAME
    NATURAL JOIN OWNS
  GROUP BY
    NAME_GAME
  ORDER BY COUNT(ID_USER) DESC) WHERE ROWNUM < 6;
```

*Código correspondente*

## Interface da Aplicação no APEX

Nossa ideia para a aplicação no APEX foi de não apenas disponibilizar as informações da base de dados, como também simular o uso dela para um usuário cadastrado na mesma. Dessa forma, fizemos uma organização visando uma UI simples de utilizar e completa, distribuídas nas 11 páginas da aplicação.

## Login



O ecrã padrão de login para a Game Platform, ele corre a autenticação padrão do APEX.

## Seleção de Utilizador

A screenshot of a user selection interface titled "Select User". At the top, there are links for "Create User", "Change User", and "Exit APP". The main area shows a list of users with their names listed vertically. The users are: Amorinus, BaronM, Bookworm, DBro, DoeJohn, EmilyProfD, Fashionista, GOAT, GuitarHero, and Inbox. Each name is preceded by a small blue link icon.

Este é o ecrã inicial da nossa aplicação. Após o login no aplicativo, a seleção de utilizador é onde se define qual dos utilizadores cadastrados é o atual usuário da plataforma, a lista de usuários é imprimida com:

```
select ID_USER from GAME_USER;
```

Já na tela de seleção dos usuários também temos acesso à primeira barra de navegação, que permite sair da aplicação (Deslogar da conta APEX), trocar o usuário (retornar para a página inicial de seleção de usuário) e criar um novo.

The screenshot shows the 'CREATE NEW USER' page. At the top, there is a navigation bar with links for 'Create User', 'Change User', and 'Exit APP'. Below the navigation bar, the page title 'Criação de Utilizador' is displayed. In the center, there is a large right-pointing arrow icon above the text 'CREATE NEW USER'. The form itself consists of several input fields: 'Choose a Username', 'Insert Your Name', 'Insert Your Email', 'Password', 'Date of Birth' (with a calendar icon), 'Insert Your Address', and 'Are you a Developer?' (with 'Yes' and 'No' buttons). Below the address field, there is a field for 'Insert Your Payment Method'. At the bottom of the form is a blue button labeled 'Create User'.

Ao selecionar o 'Create User' na barra de navegação principal, o utilizador é redirecionado para a página de cadastro. O utilizador preenche as informações e o procedimento CREATE\_USER (já explicitado acima na secção do SQL) é executado ao pressionar o botão 'Create User' da seguinte forma.

```
BEGIN
    CREATE_USER(
        :USER_ID,
        :USER_NAME,
        :USER_EMAIL,
        :USER_PASSWORD,
        :USER_ADDRESS,
        :USER_NIF,
        TO_TIMESTAMP(:USER_BIRTH, 'YYYY-MM-DD'),
        :USER_PAYMETHOD
    );
END;
```

Sendo assim, os parâmetros recebidos para a execução são os nomes dos itens do APEX que recebem as informações inseridas pelo utilizador. Um exemplo da utilização correta do procedimento é a possibilidade de diferenciar um Player de um Developer, pois como já

demonstrado anteriormente, o valor de `:USER_NIF` e `:USER_PAYMETHOD` podem ser nulos, no APEX é selecionado a opção de valores não obrigatórios. A opção de inserir o NIF só aparece quando a checkbox ‘Are you a Developer’ corresponder a ‘Yes’.

Após a criação de usuário, o utilizador é redirecionado para a página de seleção de usuário com a lista atualizada. Onde já é possível selecionar o usuário criado.

## Página All Games (Home)

The screenshot shows a user interface for managing games. At the top, there's a navigation bar with links for 'Create User', 'Change User', and 'Exit APP'. Below the navigation bar is a header titled 'All Games' with sub-links 'What's New', 'My Account', and 'Developer Manager'. The main content area is titled 'Home' and contains a table listing various video games. The table has columns for 'Name game', 'Desc game', 'Release date', 'Current price', and 'Game page'. Each game entry includes a 'VIEW MORE' button. The games listed are: Foamstars, Minecraft 2, Stardew Valley 2, The Elder Scrolls VI, The Witcher 3: Wild Hunt, Halo Infinite, God of War, Resident Evil 9, Assassins Creed 2 Remake, Final Fantasy VII Remake, and Pokémon Sword and Shield.

Name game	Desc game	Release date	Current price	Game page
Foamstars	A copy of Splatoon	15-JUN-16	59.99	<a href="#">VIEW MORE</a>
Minecraft 2	Much more pixels and a new way of seeing things, experience surviving in a new blocky world	20-SEP-16	39.99	<a href="#">VIEW MORE</a>
Stardew Valley 2	Now you can make that thing	25-JUL-17	24.99	<a href="#">VIEW MORE</a>
The Elder Scrolls VI	Embark on a new epic adventure in the world of Tamriel	11-NOV-18	49.99	<a href="#">VIEW MORE</a>
The Witcher 3: Wild Hunt	Embark on a dark and immersive journey as Geralt of Rivia in this epic RPG	19-MAY-15	29.99	<a href="#">VIEW MORE</a>
Halo Infinite	Join the Master Chief in his latest adventure against the forces of the Covenant	08-DEC-21	59.99	<a href="#">VIEW MORE</a>
God of War	Embark on a journey with Kratos in this epic action-adventure game	20-APR-18	49.99	<a href="#">VIEW MORE</a>
Resident Evil 9	Survive the horrors in a new chapter of the acclaimed Resident Evil series	09-SEP-22	39.99	<a href="#">VIEW MORE</a>
Assassins Creed 2 Remake	Relive again the beginning of Ezio story, listen to the classic music	15-NOV-19	29.99	<a href="#">VIEW MORE</a>
Final Fantasy VII Remake	Relive the classic RPG with enhanced graphics and new gameplay mechanics	10-APR-20	59.99	<a href="#">VIEW MORE</a>
Pokémon Sword and Shield	Embark on a new Pokémon adventure in the Galar region	15-NOV-19	49.99	<a href="#">VIEW MORE</a>

Após selecionar o usuário, a página que lista todos os jogos registrados é aberta. Ela é composta por um Breadcrumb inicial e um Interactive Report que faz a listagem da seguinte forma:

```
SELECT CODE_GAME,
NAME_GAME,
DESC_GAME,
RELEASE_DATE,
CURRENT_PRICE,
'VIEW' AS GAME_PAGE
FROM GAME;
```

É possível visitar a página de cada jogo ao clicar no botão 'View More', na coluna GAME\_PAGE. O botão em si é uma coluna do tipo link, a sua aparência como botão é devida ao código CSS nele presente, esta decisão é puramente cosmética e não influencia em nada o funcionamento da página. A coluna CODE\_GAME é escondida, e sua existência se deve à necessidade de passar o código de um jogo para a sua página ao clicar no botão.

## Página do Jogo (Game Info)

**GAME-PLATFORM**

All Games What's New My Account Developer Manager Create User Change User Exit APP

Game Info

Home \ Game Info

**Info about MINECRAFT 2**

Name: Minecraft 2  
Description: Much more pixels and a new way of seeing things  
Release date: 20-SEP-16  
Price: 39.99

**BUY GAME**

**Developers and Publishers**

Studio(s): Mojang Studios  
Publisher: Xbox Game Studios

**Genres:**

Action  
Adventure  
Survival

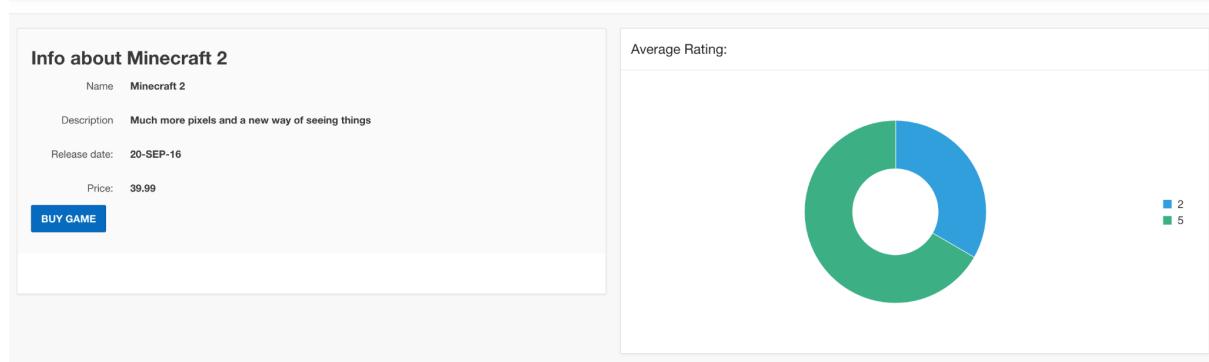
**Reviews:**

User	Review Date	Rating	Review Text
Amorinus	30-MAY-23 06:44:53.041000 PM	5	Been waiting so long for this
MasterChef	30-MAY-23 06:46:18.587000 PM	2	I LOVE THIS GAME
Inbox	30-MAY-23 06:45:44.713000 PM	5	I LOVE THIS GAME

**All active sessions:**

User	Start Time	End Time
MasterChef	30-MAY-23 06:46:30.000000 PM	-

A página do jogo é acessada somente após o usuário clicar em 'View More' na home page. Ela é composta por sete componentes diferentes e um breadcrumb que referencia a Home.



1 - A primeira componente trata-se das informações do jogo selecionado (nome, descrição, data de lançamento e preço), esses dados são recebidos da página anterior através de uma ação de link associada ao botão 'View More' da seguinte forma:

### Link Builder - Target

Target

Type: Page in this application

Page: 2

Set Items

Name	Value
P2_CODE	#CODE_GAME#
P2_NAME	#NAME_GAME#
P2_DATE	#RELEASE_DATE#
P2_PRICE	#CURRENT_PRICE#

Esse componente também possui um botão ‘Buy Game’ que corre o *PURCHASE\_GAME* caso o utilizador for um player que não possui o jogo.

### Info about God of War

Name: God of War

Description: Embark on a journey with Kratos in this epic action-adventure game

Release date: 20-APR-18

Price: 49.99

**BUY GAME**

```
DECLARE
PROPER_NAME VARCHAR2(30);
BEGIN
SELECT ID_USER INTO PROPER_NAME
FROM GAME_USER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
PURCHASE_GAME(:P2_CODE, PROPER_NAME, SYSDATE);
END;
```

2 - A segunda componente é a média das reviews realizadas, que executa a seguinte Query:

Average Rating:



■ 1  
■ 2  
■ 3  
■ 5

```
SELECT rating, COUNT(*) AS count
FROM reviews
WHERE code_game = :P2_CODE
GROUP BY rating
```

3 - A terceira componente é a informação de quem desenvolveu o jogo, onde se diferencia jogos desenvolvidos por estúdios ou desenvolvidos por developers.

### Developers and Publishers

Studio(s): **Mojang Studios**

Publisher: **Xbox Game Studios**

*Exemplo desenvolvedores de ‘Minecraft 2’*

## Developers and Publishers

Developer(s): **John Doe:Michael Johnson**

*Exemplo desenvolvedores de 'Mythical Realms'*

As Querys necessárias para essa componente que necessita verificar o nome dos estúdios, nome do developer, nome da publisher são:

```
select GAME_USER.NAME_USER as NAME_USER
from GAME_USER GAME_USER,
DEVELOPER DEVELOPER,
DEV_CREATES DEV_CREATES
where DEV_CREATES.ID_USER=DEVELOPER.ID_USER
and DEVELOPER.ID_USER=GAME_USER.ID_USER
and DEV_CREATES.CODE_GAME = :P2_CODE
```

```
SELECT
DISTINCT CP.NAME_COMPANY
FROM
STUDIO S INNER JOIN COMPANY CP ON S.NIF_COMPANY = CP.NIF_COMPANY
INNER JOIN CREATES C
ON S.NIF_COMPANY = C.NIF_STUDIO
WHERE
C.CODE_GAME = :P2_CODE
UNION
SELECT
DISTINCT CP.NAME_COMPANY
FROM
STUDIO S INNER JOIN COMPANY CP ON S.NIF_COMPANY = CP.NIF_COMPANY
INNER JOIN HAS_PARTNERS HP
ON S.NIF_COMPANY = HP.NIF_COMPANY
WHERE
HP.CODE_GAME = :P2_CODE;
```

```
select COMPANY.NAME_COMPANY as NAME_COMPANY
from COMPANY COMPANY,
PUBLISHER PUBLISHER,
PUBLISHES PUBLISHES
where PUBLISHES.NIF_PUBLISHER=PUBLISHER.NIF_COMPANY
and PUBLISHER.NIF_COMPANY=COMPANY.NIF_COMPANY
and PUBLISHES.CODE_GAME = :P2_CODE
```

4. A componente de informação dos gêneros do jogo, um interactive report onde o nome dos gêneros são links para uma outra página específica para gêneros, segue a Query:

Genres:

Q ▾		Go	Actions ▾
Name			
Action			
Adventure			
Third-Person-Shooter			
Horror			

```
select NAME_GENRE AS Name,
CODE_GENRE AS Code
FROM GENRE NATURAL JOIN HAS_GENRE
WHERE CODE_GAME = :P2_CODE;
```

5. A componente das reviews, onde existe a listagem das reviews e a possibilidade de atualizar ou escrever uma caso o utilizador possua o jogo.

**Reviews:**

Para listagem e para a ação realizada ao clicar no botão ‘Send’ foram executados os seguintes comandos, respectivamente:

```
SELECT ID_USER,
REVIEW_TEXT,
RATING,
REVIEW_DATE
FROM REVIEWS
WHERE CODE_GAME = :P2_CODE;
```

```
DECLARE
    REAL_ID VARCHAR2(100);
BEGIN
    SELECT
        ID_USER INTO REAL_ID
    FROM
        GAME_USER
    WHERE
        UPPER(ID_USER) = UPPER(:P0_USER);
    -- Check if a review entry already exists for the user and code
    DECLARE
        REVIEW_NUM NUMBER;
    BEGIN
        SELECT
            COUNT(*) INTO REVIEW_NUM
        FROM
            REVIEWS
        WHERE
            ID_USER = REAL_ID
            AND CODE_GAME = :P2_CODE;
        IF REVIEW_NUM > 0 THEN
            UPDATE REVIEWS
            SET
                RATING = :P2_STARS,
                REVIEW_TEXT = :P2_REVIEW,
                REVIEW_DATE = CURRENT_TIMESTAMP
            WHERE
                ID_USER = REAL_ID
                AND CODE_GAME = :P2_CODE;
        ELSE
            INSERT INTO REVIEWS VALUES (
                REAL_ID,
                :P2_CODE,
                CURRENT_TIMESTAMP,
                :P2_STARS,
                :P2_REVIEW
            );
        END IF;
    END;
END;
```

6. Componente de sessões, interactive report com botões de iniciar uma sessão no jogo da página e encerrar uma sessão corrente do jogador.

All active sessions:		
User	Start Time	End Time
MasterChef	30-MAY-23 06.46.30.000000 PM	30-MAY-23 06.55.51.201000 PM
GuitarHero	30-MAY-23 08.14.30.000000 PM	-

A listagem das sessões e a implementação dos botões são realizadas da seguinte forma:

```

        SELECT ID_USER, DATE_START, DATETIME_END
        FROM GAME_SESSION
        WHERE CODE_GAME = :P2_CODE;

DECLARE
    REAL_ID VARCHAR2(100);
BEGIN
    SELECT
        ID_USER INTO REAL_ID
    FROM
        GAME_USER
    WHERE
        UPPER(ID_USER) = UPPER(:P0_USER);
    ADD_SESSION(REAL_ID, :P2_CODE, SYSDATE, NULL); END_SESSION(REAL_ID);
END;

```

```

DECLARE
    REAL_ID VARCHAR2(100);
BEGIN
    SELECT
        ID_USER INTO REAL_ID
    FROM
        GAME_USER
    WHERE
        UPPER(ID_USER) = UPPER(:P0_USER);
    END;

```

7. Componente Servers, aparição condicional do caso do jogo ser Multiplayer

Servers:		
<input type="text" value="Insert Server ID"/> <input type="button" value="Join"/> <span style="float: right;"><input type="button" value="Create"/></span>		
Region	Server ID	Created On
US	1	30-MAY-23 02.18.13.000000 PM
EU	2	30-MAY-23 02.18.13.000000 PM

A listagem e a condição para a componente aparecer são o seguinte código:

```

SELECT REGION, SERVER_ID, DATETIME_CREATION
FROM GAME_SERVER
WHERE CODE_GAME = :P2_CODE;

```

```

SELECT *
FROM MULTIPLAYER_GAME
WHERE CODE_GAME = :P2_CODE;

```

O botão Join Server e Quit Server são semelhantes à implementação do Start Session (com o *ADD\_SESSION* agora recebendo o *SERVER\_ID*) e End Session, justamente por se tratarem de uma sessão multiplayer, logo a informação aparecerá na tabela acima de sessões. O código dos dois botões é o seguinte:

```

DECLARE
    REAL_ID VARCHAR2(100);
BEGIN
    SELECT
        ID_USER INTO REAL_ID
    FROM
        GAME_USER
    WHERE
        UPPER(ID_USER) = UPPER(:P0_USER);
    ADD_SESSION(REAL_ID, :P2_CODE, SYSDATE, :P2_SERV_ID);    END_SESSION(REAL_ID);
END;

```

```

DECLARE
    REAL_ID VARCHAR2(100);
BEGIN
    SELECT
        ID_USER INTO REAL_ID
    FROM
        GAME_USER
    WHERE
        UPPER(ID_USER) = UPPER(:P0_USER);
    END_SESSION(REAL_ID);
END;

```

O ‘Create’ executa o *ADD\_SERVER* passando os argumentos inseridos de Região e Host Key, da seguinte forma:

```

DECLARE
    REAL_ID VARCHAR2(100);
BEGIN
    SELECT
        ID_USER INTO REAL_ID
    FROM
        GAME_USER
    WHERE
        UPPER(ID_USER) = UPPER(:P0_USER);

    ADD_SERVER(:P2_CODE, SYSDATE, :P2_SERV_REGION, REAL_ID, :P2_SERV_HOSTK);
END;

```

# Página do Gênero (Genre Games)

Home \ Game Info \

## Genre Games

These are our First-Person-Shooter games

**Studio Titles:**

Studio	Release date	Current price	Average rating	Name game	Desc game
Bungie	09-DEC-22	49.99	No reviews yet	Destiny 3	Embark on an epic sci-fi adventure in the next installment of the Destiny franchise
Blizzard Entertainment	24-MAY-16	29.99	No reviews yet	Overwatch	Engage in team-based multiplayer battles as diverse heroes with unique abilities
343 Industries	08-DEC-21	59.99	No reviews yet	Halo Infinite	Join the Master Chief in his latest adventure against the forces of the Covenant
CD Projekt Red	17-SEP-22	39.99	3	Cyberpunk 2077	Immerse yourself in a futuristic open world filled with high-tech gadgets and cybernetic enhancements
343 Industries	28-OCT-22	59.99	No reviews yet	Mass Effect: Reborn	Take command of the Normandy once again and lead the fight against a new extraterrestrial threat
Blizzard Entertainment	04-DEC-20	39.99	No reviews yet	BioShock: Beyond the Abyss	Plunge into the depths of a new underwater city in this atmospheric first-person shooter
Treyarch	12-NOV-21	39.99	No reviews yet	Call of Duty: Black Ops II Remastered	Experience the iconic first-person shooter with improved graphics and enhanced gameplay

1 - 7

**Independent titles:**

Name game	Release date	Current price	Average rating	Desc game
Cybernetic Warfare	25-NOV-19	59.99	No reviews yet	Command a squad of advanced cybernetic soldiers in a futuristic war against rogue AI
Team Fortress 2	17-SEP-22	0	No reviews yet	The most fun you can have online!

1 - 2

Esta página é acessada a partir de uma página de jogo, ela é composta por um Breadcrumb que referencia a página de jogo usada e a home, dois componentes que são condicionais, que informam quais jogos registrados possuem esse mesmo gênero.

A primeira tabela é um interactive report de jogos desenvolvidos por estúdios, segue a correspondente Query e condição para aparecer:

```

SELECT
    G.NAME_GAME,
    G.CODE_GAME,
    P.NAME_COMPANY AS STUDIO,
    G.RELEASE_DATE,
    G.CURRENT_PRICE,
    G.DESC_GAME,
    COALESCE(TO_CHAR(AVG(R.RATING)), 'No reviews yet') AS AVERAGE_RATING
FROM
    HAS_GENRE H JOIN GAME G ON H.CODE_GAME = G.CODE_GAME
    JOIN CREATES C ON G.CODE_GAME = C.CODE_GAME
    JOIN COMPANY P ON P.NIF_COMPANY = C.NIF_STUDIO
    LEFT JOIN OWNS O ON G.CODE_GAME = O.CODE_GAME
    LEFT JOIN REVIEWS R ON G.CODE_GAME = R.CODE_GAME
WHERE
    H.CODE_GENRE = :P16_GENRE
GROUP BY
    G.NAME_GAME,
    G.CODE_GAME,
    P.NAME_COMPANY,
    G.RELEASE_DATE,
    G.DESC_GAME,
    G.CURRENT_PRICE;
  
```

```

SELECT
    G.CODE_GAME,
    P.NAME_COMPANY AS STUDIO,
    G.RELEASE_DATE,
    G.CURRENT_PRICE,
    COALESCE(TO_CHAR(AVG(R.RATING)), 'No reviews yet') AS AVERAGE_RATING
FROM
    HAS_GENRE H JOIN GAME G ON H.CODE_GAME = G.CODE_GAME
    JOIN CREATES C ON G.CODE_GAME = C.CODE_GAME
    JOIN COMPANY P ON P.NIF_COMPANY = C.NIF_STUDIO
    LEFT JOIN OWNS O ON G.CODE_GAME = O.CODE_GAME
    LEFT JOIN REVIEWS R ON G.CODE_GAME = R.CODE_GAME
WHERE
    H.CODE_GENRE = :P16_GENRE
GROUP BY
    G.CODE_GAME,
    P.NAME_COMPANY,
    G.RELEASE_DATE,
    G.CURRENT_PRICE;
  
```

A segunda tabela é um interactive report de jogos desenvolvidos por developers, segue a correspondente Query e condição para aparecer:

```
SELECT
    G.CODE_GAME,
    G.NAME_GAME,
    G.RELEASE_DATE,
    G.CURRENT_PRICE,
    G.DESC_GAME,
    COALESCE(TO_CHAR(AVG(R.RATING)), 'No reviews yet') AS AVERAGE_RATING
FROM
    HAS_GENRE H JOIN GAME G ON H.CODE_GAME = G.CODE_GAME
    JOIN DEV_CREATES C ON G.CODE_GAME = C.CODE_GAME
    JOIN GAME_USER U ON U.ID_USER = C.ID_USER
    LEFT JOIN OWNS O ON G.CODE_GAME = O.CODE_GAME
    LEFT JOIN REVIEWS R ON G.CODE_GAME = R.CODE_GAME
WHERE
    H.CODE_GENRE = :P16_GENRE
GROUP BY
    G.CODE_GAME,
    G.NAME_GAME,
    G.RELEASE_DATE,
    G.DESC_GAME,
    G.CURRENT_PRICE;

SELECT
    G.CODE_GAME,
    G.NAME_GAME,
    G.RELEASE_DATE,
    G.CURRENT_PRICE,
    COALESCE(TO_CHAR(AVG(R.RATING)), 'No reviews yet') AS AVERAGE_RATING
FROM
    HAS_GENRE H JOIN GAME G ON H.CODE_GAME = G.CODE_GAME
    JOIN DEV_CREATES C ON G.CODE_GAME = C.CODE_GAME
    JOIN GAME_USER U ON U.ID_USER = C.ID_USER
    LEFT JOIN OWNS O ON G.CODE_GAME = O.CODE_GAME
    LEFT JOIN REVIEWS R ON G.CODE_GAME = R.CODE_GAME
WHERE
    H.CODE_GENRE = :P16_GENRE
GROUP BY
    G.CODE_GAME,
    G.NAME_GAME,
    G.RELEASE_DATE,
    G.CURRENT_PRICE;
```

## Página What's New

Esta é uma página que contém informações dos jogos mais vendidos da plataforma, os jogos mais bem avaliados e os jogos mais jogados no momento.

Segue a Query para pegar os Top 5 jogos da plataforma:

```
SELECT * FROM (SELECT
    NAME_GAME AS GAME,
    COUNT(ID_USER) AS SALES
FROM
    GAME
    NATURAL JOIN OWNS
GROUP BY
    NAME_GAME
ORDER BY COUNT(ID_USER) DESC) WHERE ROWNUM < 6;
```

Segue a Query para o Top 5 Jogos mais bem avaliados da Plataforma:

```
SELECT * FROM (SELECT
    NAME_GAME AS GAME,
    AVG(RATING) AS RATING
FROM
    GAME
    NATURAL JOIN OWNS NATURAL JOIN REVIEWS
GROUP BY
    NAME_GAME
ORDER BY RATING DESC) WHERE ROWNUM < 6;
```

Segue a Query para o Top 5 Jogos mais jogados no momento:

```
SELECT * FROM (SELECT
    NAME_GAME AS GAME,
    COUNT(ID_USER) AS PLAYERS
FROM
    GAME
    NATURAL JOIN GAME_SESSION
WHERE
    DATETIME_END IS NULL
GROUP BY
    NAME_GAME
ORDER BY COUNT(ID_USER) DESC) WHERE ROWNUM < 6;
```

# Página do Utilizador (My Account)

The screenshot shows the 'My Account' section of a game platform. At the top, there's a navigation bar with links for 'All Games', 'What's New', 'My Account' (which is highlighted in blue), and 'Developer Manager'. Below the navigation is a header titled 'Inbox's Info:'.

**User Information:**

Username	Inbox	Email	inboxmoss@gmail.com
Full Name	Lucas Geraldo	Date of Birth	DECEMBER 15 1990
Address	Lisbon - Portugal		
Payment Method	PayPal		

Buttons for 'Update info' and 'Change user' are at the bottom of this section.

**Games you own:**

Name game	Purchase price	Purchase date	Purchase method
Minecraft 2	99.99	30-MAY-23 06:45:33.000000 PM	PayPal
Cyberpunk 3077	99.99	30-MAY-23 02:18:12.000000 PM	PayPal

Page navigation: 1 - 2

**Reviews you have written:**

Name game	Review date	Rating	Review text
Minecraft 2	30-MAY-23 06:45:44.713000 PM	5	I LOVE THIS GAME
Cyberpunk 3077	30-MAY-23 02:18:13.193000 PM	1	I am inbox

Page navigation: 1 - 2

Ativar o Windows. Acesse Configurações para ativar o Windows.

Página centrada no usuário, responsável por mostrar informação sobre ele, assim como permitir alterações em seus dados e a troca de usuário. Está dividida em 4 secções principais:

## 1. O painel de informações do usuário:

This screenshot shows the 'Inbox's Info:' panel for a user named 'Amorinus'. It displays basic user details and payment method information.

**User Information:**

Username	Inbox	Email	inboxmoss@gmail.com
Full Name	Lucas Geraldo	Date of Birth	DECEMBER 15 1990
Address	Lisbon - Portugal		
Payment Method	PayPal		

Buttons for 'Update info' and 'Change user' are at the bottom.

Mostra as várias informações do usuário (tabela `game_user`) assim como informações específicas de um jogador (método de pagamento, como no exemplo acima),

This screenshot shows the 'Amorinus's Info:' panel for a developer named 'Amorinus'. It displays detailed developer information, including employer and NIF.

**Developer Information:**

Username	Amorinus	Email	ph.terceiro@gmail.com
Full Name	Pedro Henrique	Date of Birth	JULY 10 1996
Address	Santa Catarina - Brasil		
Payment Method	Debit	NIF	174345457
Employer	Bethesda Game Studios		

Buttons for 'Update info' and 'Change user' are at the bottom.

ou mostrar informações específicas do desenvolvedor (seu nif, como na imagem acima).

Cada um dos campos de informação desta seção tem a sua própria query no SQL:

```
SELECT ID_USER
FROM GAME_USER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

```
SELECT EMAIL
FROM GAME_USER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

```
SELECT NAME_USER
FROM GAME_USER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

```
SELECT USER_ADDRESS
FROM GAME_USER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

```
SELECT TO_CHAR(DATE_OF_BIRTH, 'MONTH DD YYYY') AS READABLE_DATE_OF_BIRTH
FROM GAME_USER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

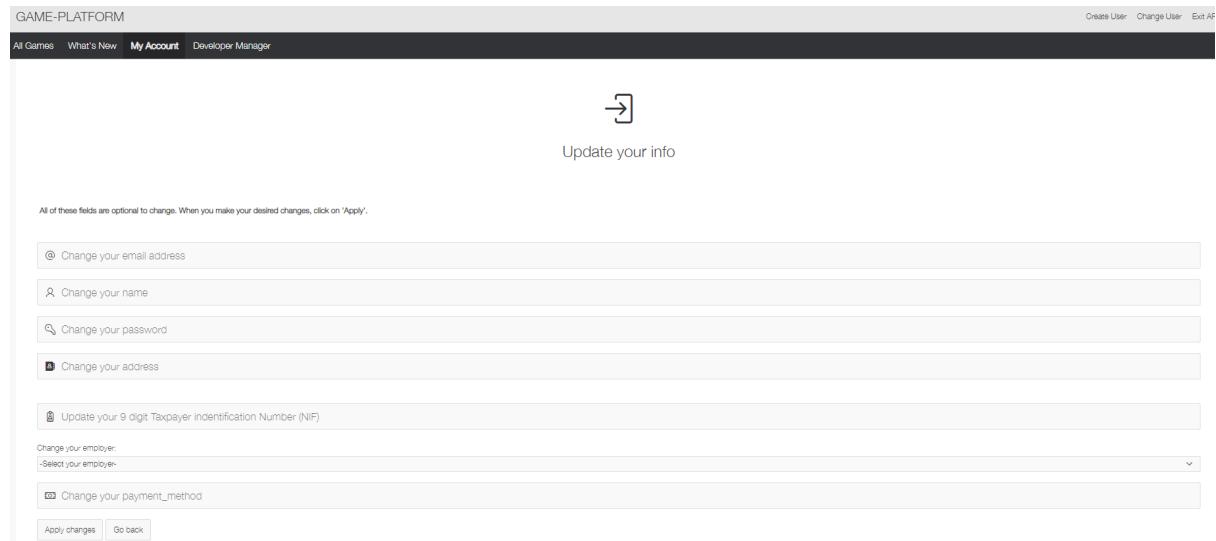
---

```
SELECT PAYMENT_METHOD
FROM PLAYER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

```
SELECT NIF_DEVELOPER
FROM DEVELOPER
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

```
SELECT NAME_COMPANY
FROM WORKS NATURAL JOIN GAME_USER NATURAL JOIN COMPANY
WHERE UPPER(ID_USER) = UPPER(:P0_USER);
```

Está também presente dois botões. "Change User" tem a mesma funcionalidade do botão de mesmo nome no topo da página, leva para a página de seleção de usuário. "Update Info" leva a uma outra página:



The screenshot shows a user interface for updating account information. At the top, there's a navigation bar with links for 'All Games', 'What's New', 'My Account' (which is highlighted in blue), and 'Developer Manager'. On the far right of the bar are 'Create User', 'Change User', and 'Exit AF' buttons. Below the navigation bar, there's a large button with a right-pointing arrow containing the text 'Update your info'. Underneath this button, a note says 'All of these fields are optional to change. When you make your desired changes, click on 'Apply''. There are five input fields with labels: 'Change your email address' (with an '@' icon), 'Change your name' (with a person icon), 'Change your password' (with a lock icon), 'Change your address' (with a house icon), and 'Update your 9 digit Taxpayer identification Number (NIF)' (with a document icon). Below these fields is a dropdown menu labeled 'Change your employer' with the placeholder '-Select your employer-'. At the bottom of the form are two buttons: 'Apply changes' and 'Go back'.

Esta página possui vários campos de preenchimento de dados. Quando o botão “Apply Changes” é clicado, aqueles que tiveram algum dado inserido terão os seus respectivos dados no perfil de usuário alterados para o dado inserido. Para os campos que se mantiveram em branco quando o botão foi apertado, a sua respectiva informação do usuário não é alterada. O botão “Go Back” retorna para a página do utilizador.

Quando o usuário não é do tipo desenvolvedor, a página mostra um seletor:

Quando “Yes” é selecionado, aparece o campo para preencher o NIF de desenvolvedor que, se é preenchido e o botão “Apply Changes” é pressionado, o usuário é registado como desenvolvedor também.

2. A página de visualização dos jogos que este usuário desenvolveu de forma independente/trabalhando para um estúdio.

Independent Titles you have worked on

Name game	Release date	Current price	Sold	Average rating
Steampunk Chronicles	15-JUL-16	39.99	0	No reviews yet

1 - 1

Studio Titles you have worked on

Name game	Release date	Current price	Sold	Average rating	Studio
The Elder Scrolls VI	11-NOV-18	49.99	0	No reviews yet	Bethesda Game Studios
Fallout: New Frontiers	28-APR-23	49.99	0	No reviews yet	Bethesda Game Studios
World of Warcraft: The Next Chapter	17-MAY-22	59.99	0	No reviews yet	Bethesda Game Studios

Ativar o Windows  
Acesse Configurações para ativar o Windows. 1 - 3

Sendo o Report acima os títulos independentes e abaixo os títulos de estúdio. Ambas os reports seguem, respetivamente, as seguintes queries:

```
SELECT
    G.NAME_GAME,
    G.RELEASE_DATE,
    G.CURRENT_PRICE,
    COUNT(W.CODE_GAME) AS SOLD,
    COALESCE(TO_CHAR(ROUND(AVG(R.RATING))), 'No reviews yet') AS AVERAGE_RATING
FROM
    GAME G
    JOIN DEV_CREATES O ON G.CODE_GAME = O.CODE_GAME
    LEFT JOIN OWNS W ON W.CODE_GAME = G.CODE_GAME
    LEFT JOIN REVIEWS R ON R.CODE_GAME = G.CODE_GAME
WHERE
    UPPER(O.ID_USER) = UPPER(:P0_USER)
GROUP BY
    G.CODE_GAME, G.NAME_GAME, G.RELEASE_DATE, G.CURRENT_PRICE;

SELECT
    G.NAME_GAME,
    P.NAME_COMPANY AS STUDIO,
    G.RELEASE_DATE,
    G.CURRENT_PRICE,
    COUNT(O.CODE_GAME) AS SOLD,
    COALESCE(TO_CHAR(ROUND(AVG(R.RATING))), 'No reviews yet') AS AVERAGE_RATING
FROM
    WORKS W
    JOIN CREATES C ON W.NIF_COMPANY = C.NIF_STUDIO
    JOIN GAME G ON C.CODE_GAME = G.CODE_GAME
    JOIN COMPANY P ON C.NIF_STUDIO = P.NIF_COMPANY
    LEFT JOIN OWNS O ON G.CODE_GAME = O.CODE_GAME
    LEFT JOIN REVIEWS R ON G.CODE_GAME = R.CODE_GAME
WHERE
    UPPER(W.ID_USER) = UPPER(:P0_USER)
GROUP BY
    G.NAME_GAME,
    P.NAME_COMPANY,
    G.RELEASE_DATE,
    G.CURRENT_PRICE;
```

### 3. A página de visualização dos jogos que este usuário comprou.

Games you own			
Name game	Purchase price	Purchase date	Purchase method
Minecraft 2	39.99	30-MAY-23 08:48:33.000000 PM	PayPal
Cyberpunk 2077	39.99	30-MAY-23 02:18:12.000000 PM	PayPal
1 - 2			

Mostra as informações relativas à compra de cada jogo, como o preço, data e método de pagamento na altura da compra. Segue a seguinte query:

```
SELECT g.NAME_GAME, o.PURCHASE_DATE, o.PURCHASE_METHOD,
       o.PURCHASE_PRICE
  FROM GAME g
 JOIN OWNS o ON g.CODE_GAME = o.CODE_GAME
 WHERE UPPER(o.ID_USER) = UPPER(:P0_USER);
```

### 4. Informação sobre as reviews deste usuário.

Reviews you have written			
Name game	Review date	Rating	Review text
Minecraft 2	30-MAY-23 08:45:44.713000 PM	5	I LOVE THIS GAME
Cyberpunk 2077	30-MAY-23 02:18:13.133000 PM	1	i am inbox
1 - 2			

Ilustra as informações sobre cada review que este usuário fez, sendo estas o nome do jogo, a data da review, a sua avaliação e o texto de avaliação. Segue a seguinte query:

```
SELECT G.NAME_GAME, R.REVIEW_DATE, R.RATING, R.REVIEW_TEXT
  FROM GAME G
 JOIN REVIEWS R ON G.CODE_GAME = R.CODE_GAME
 WHERE upper(R.ID_USER) = upper(:P0_USER);
```

# Página do Developer Manager

The screenshot shows a web-based application for managing studios and developers. At the top, there is a navigation bar with links for 'Home', 'Log in', 'Forgot password?', and 'Help'. Below the navigation, there are two main sections:

- Studios:** A table with columns for 'Name company' and 'Actions'. It lists various game studios: Rockstar North, Rockstar India, Rockstar Toronto, CAPCOM, Square Enix, Bethesda Game Studios, Arkane, Mojang Studios, 343 Industries, Bungie, Blizzard Entertainment, Treyarch, Naughty Dog, Santa Monica Studios, Game Freak, Sonic Team, Rito Games, Kojima Productions, CD Projekt Red, Redpoint, and PROJECT ACEs. A single row for 'Mojang Studios' is selected.
- Developers:** A table with columns for 'Name developer' and 'Actions'. It lists several developer names: TechMong, Total 1.

Esta página é responsável por permitir a edição, adição e remoção de desenvolvedores de qualquer estúdio, baseado em um *Master Detail*.

This screenshot shows the 'Studios' table from the previous interface. The 'Mojang Studios' row is now highlighted with a blue selection box around its entire row. The other rows are standard white. The table has columns for 'Name company' and 'Actions'.

A primeira tabela (conhecida como *Master*) é um seletor de estúdios, mostrando o nome do estúdio a selecionar. No exemplo acima o estúdio “Mojang Studios” está selecionado. Os elementos desta tabela não podem ser editados e esta segue a seguinte query:

```
select S.NIF_COMPANY, C.NAME_COMPANY  
from STUDIO S, COMPANY C  
WHERE S.NIF_COMPANY = C.NIF_COMPANY
```

Developers				Save
		Search: All Text Columns	Go	Actions
			Add Row	
	JanePlays			
	BaronM			
	SarahCRW			

Total 3

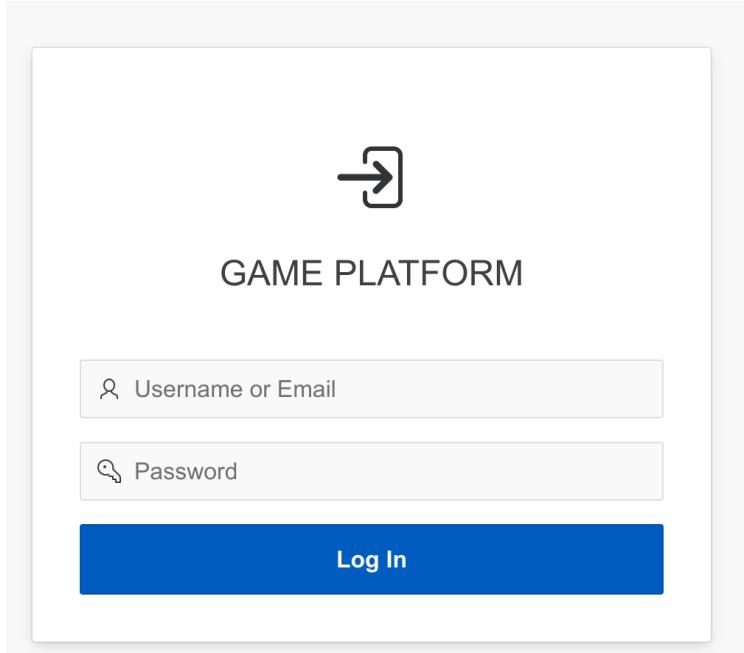
A segunda tabela (conhecida como *Detail*), mostra o ID dos desenvolvedores trabalhando no estúdio selecionado no Master e permite a sua edição para outro ID de usuário desenvolvedor válido na base de dados, a adição de outro desenvolvedor, ou a remoção completa de um tuplo. Esta tabela segue a seguinte query:

```
select W.NIF_COMPANY, W.ID_USER
from WORKS W
```

É interessante relatar que, como esta tabela tem a habilidade de alterar os dados nela presente, uma junção natural com outras tabelas (developer e game\_user por exemplo, para mostrar mais dados do desenvolvedor) não é possível. Dado que quando esta visualização alterada é transferida de volta para a base de dados, não é possível saber quais tuplos da visualização são correspondentes aos da base de dados. Logo, a informação fica perdida.

# Manual do utilizador

1. Faça login com as credenciais da sua conta APEX



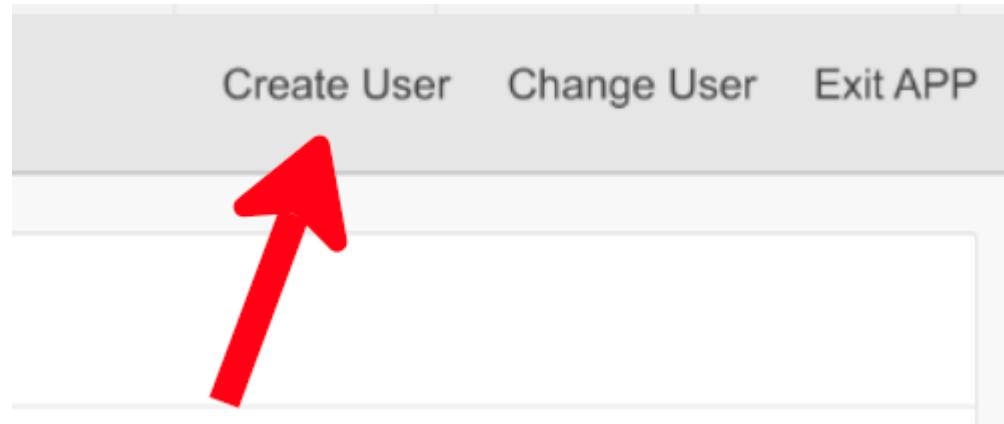
2. Selecione um utilizador da Game-Platform

- a. Caso deseje selecionar um utilizador já criado, carregue sobre o ID de um dos utilizadores da lista. Você será redirecionado à página inicial.

The image shows a screenshot of a user selection interface titled 'Select User'. At the top, there are buttons for 'Create User', 'Change User', and 'Exit APP'. Below this is a search bar with a placeholder 'Search: All Text Columns' and a 'Go' button. To the right of the search bar is a 'Reset' button. The main area displays a table with three rows of user data. The first row contains 'Amorinus', 'BaronM', and a red arrow pointing to the 'Bookworm' entry in the third row. The 'Bookworm' row is highlighted with a light gray background.

Amorinus	BaronM	Bookworm
Amorinus	BaronM	Bookworm

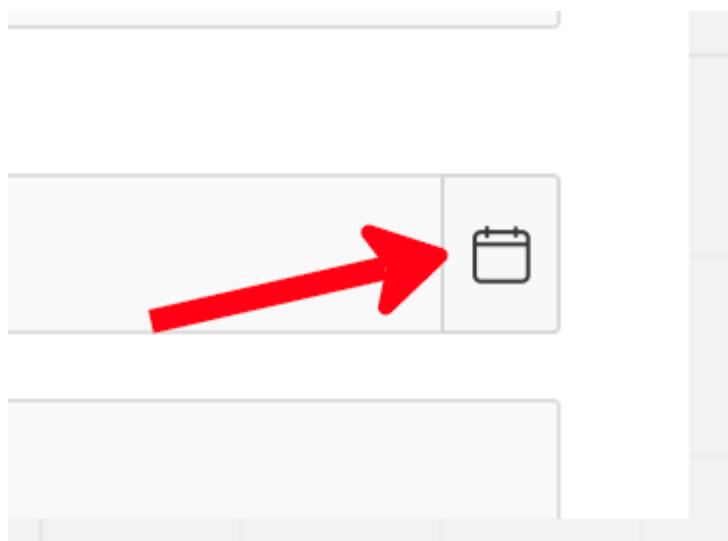
- b. Caso queira criar um novo utilizador, carregue em 'Create User' no canto superior direito do ecrã.



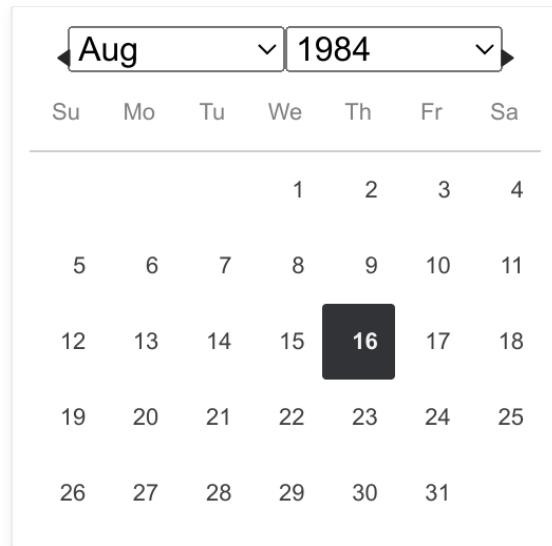
- i. Preencha o seu nome de utilizador pretendido, o seu nome completo, o seu email e a senha desejada para a conta.

- ii. Insira a sua data de nascimento seguindo estes passos:

1. No canto esquerdo da caixa 'Date of Birth', carregue neste ícone



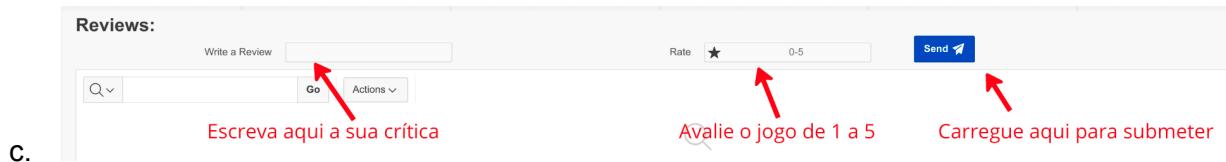
2. Deverá aparecer no ecrã um calendário.



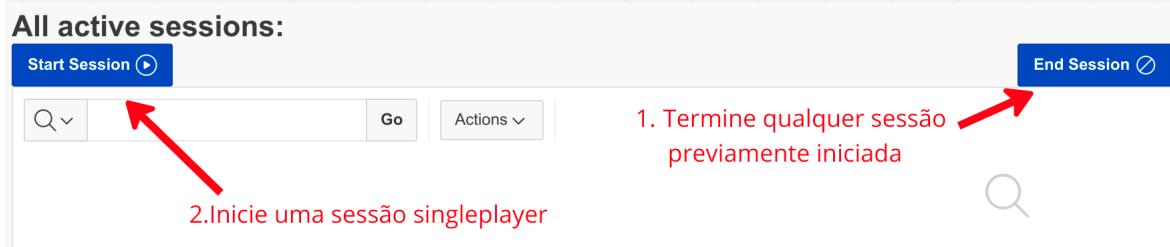
- a. Na seleção no lado superior direito selecione o seu ano de nascimento
  - b. Na seleção à esquerda escolha o seu mês de nascimento.
  - c. No calendário escolha o dia em que nasceu.
- iii. A seguir informe o seu Endereço atual
  - iv. Se você for um desenvolvedor, selecione 'Yes' na pergunta 'Are you a Developer?'
    1. Preencha com o seu NIF a caixa de texto que vem a seguir.
  - v. Insira o seu método de pagamento se for comprar jogos. Atenção: **Se você não for um desenvolvedor, esta opção é de preenchimento obrigatório.**
  - vi. Carregue em Create User.
  - vii. Você será redirecionado à página de seleção de utilizadores. Siga o passo 2.a)

3. Na página Inicial você é apresentado a uma lista de jogos. Carregue em 'View More' Em qualquer jogo para visualizar a sua página.

- Se for um jogador, para comprar um jogo carregue em 'Buy Game'
- Se tiver efetuado o passo anterior, para escrever uma avaliação do jogo, desça a página até a seção 'Reviews'

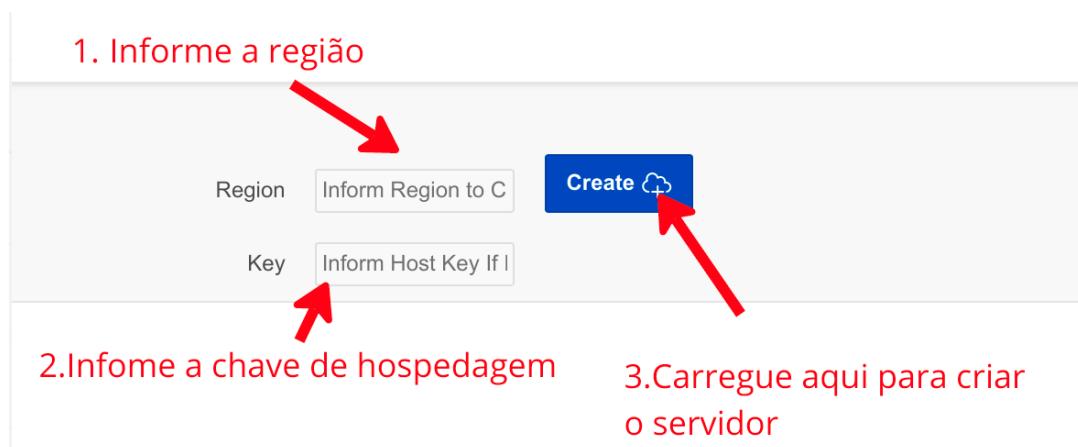


4. Para começar uma sessão singleplayer de um jogo que possua desça a página do jogo até 'All active Sessions'



5. Para começar uma sessão multiplayer desça até 'Servers' e siga estes passos:

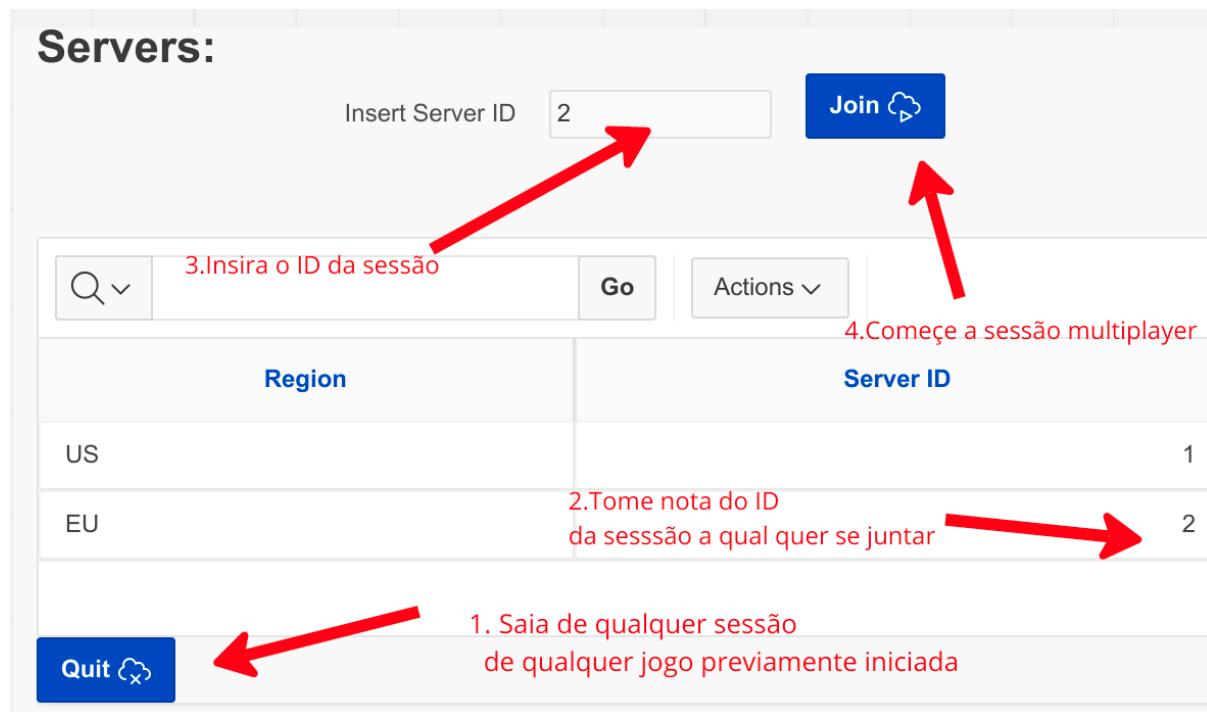
- Se quiser criar um servidor:



i.

- É importante relembrar que **qualquer jogador pode criar um servidor de qualquer jogo, independentemente de possuí-lo ou não.**

- Se quiser se juntar a um servidor de algum jogo multijogador que possua, siga estes passos:



- A navegação no site é feita pela barra principal. Para exemplificar, vamos ver algumas estatísticas interessantes em **What's new**.

The screenshot shows the 'GAME-PLATFORM' website. At the top, there is a navigation bar with tabs: 'All Games' (highlighted in white), 'What's New' (highlighted in blue), 'My Account', and 'Developer Manager'. Below the navigation bar, the word 'Home' is displayed. The main content area is titled 'The Best' and describes the top 5 rated games of all time. It includes a table with columns 'Game' and 'Rating'. The games listed are: Halo Infinite (Rating 5), Cybernetic Arena (Rating 5), Minor Pay Manual (Rating 4), Resident Evil 9 (Rating 4), and Stardew Valley 2 (Rating 3.5).

- Para ver detalhes da sua conta, vá em '**My Account**' na barra principal. Nesta página também pode ver os seus jogos, assim como aqueles publicados na plataforma.
- Para editar a sua conta, em '**My Account**' carregue em '**Update Info**'. Você será encaminhado para uma página de edição dos seus dados. **Note que aqui todos os campos são opcionais.**

All of these fields are optional to change. When you make your desired changes, click on 'Apply'.

The screenshot shows a user profile edit interface. At the top, it says "All of these fields are optional to change. When you make your desired changes, click on 'Apply'." Below are several input fields:

- Email: new@email.com
- Name: New Name
- .....: ..... (Note: Atualize todos os campos que deseja. Note que a depender do seu tipo de utilizador nem todas as opções podem estar disponíveis.)
- New City - New Country
- Phone: 0000000000

Below these fields is a section for changing the employer:

Change your employer:  
PROJECT ACES

At the bottom right of the page is a note: "Carregue aqui para aplicar as mudanças".

Buttons at the bottom left: "Apply changes" (highlighted with a red arrow) and "Go back".

9. Para gerir os desenvolvedores trabalham num estúdio, na barra principal carregue em 'Developer manager'

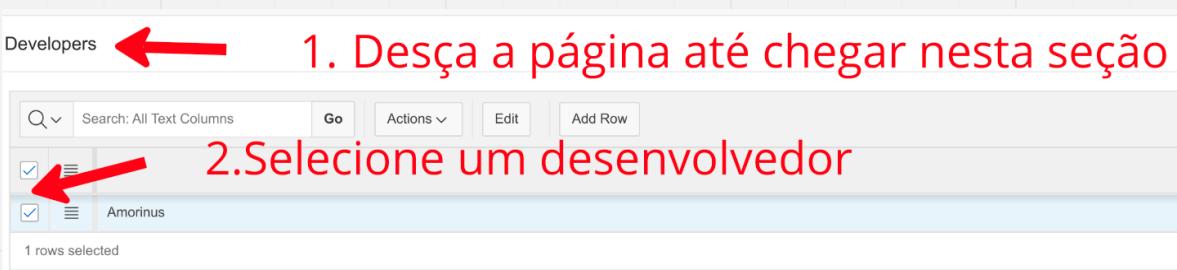
The screenshot shows the 'Studios' section of the developer manager. The title is 'Studios'. There is a search bar and an 'Actions' dropdown. The table lists studios:

		Name company
<input type="checkbox"/>	<input type="checkbox"/>	Rockstar North
<input type="checkbox"/>	<input type="checkbox"/>	Rockstar India
<input type="checkbox"/>	<input type="checkbox"/>	Rockstar Toronto
<input type="checkbox"/>	<input type="checkbox"/>	CAPCOM
<input type="checkbox"/>	<input type="checkbox"/>	Square Enix
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Bethesda Game Studios

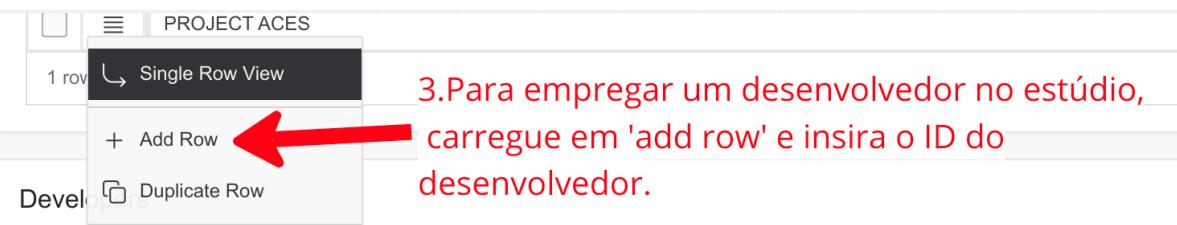
A red arrow points to the checkbox next to Bethesda Game Studios, which is highlighted with a blue border. Red text above the table says "Selecionar um estúdio".

a.

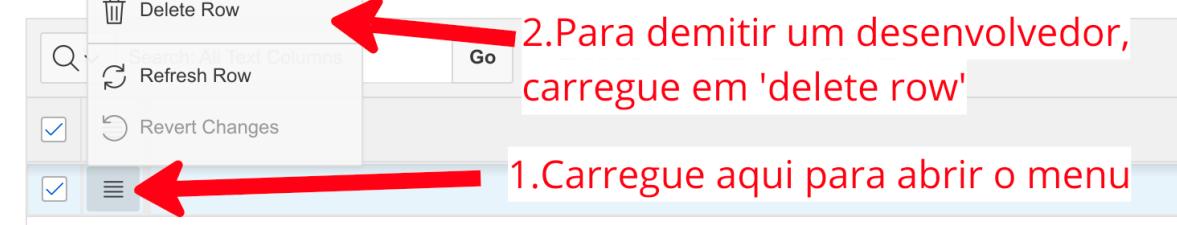
1. Desça a página até chegar nesta seção



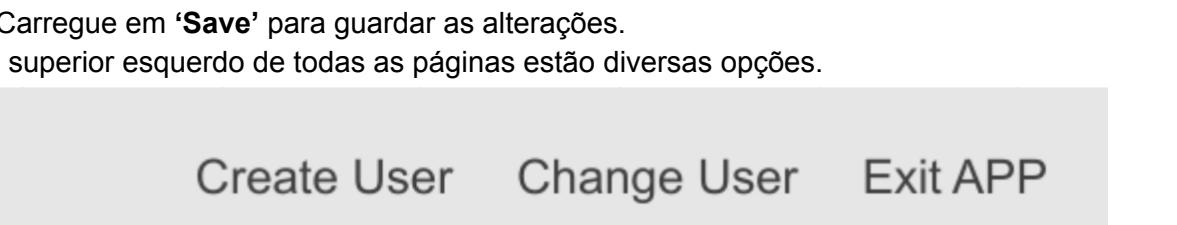
2. Selecione um desenvolvedor



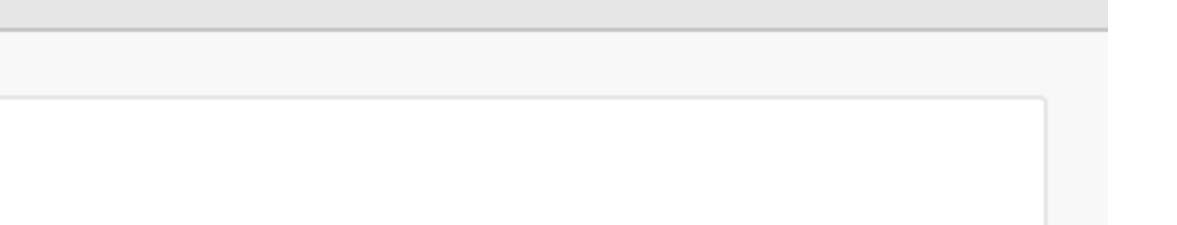
3. Para empregar um desenvolvedor no estúdio, carregue em 'add row' e insira o ID do desenvolvedor.



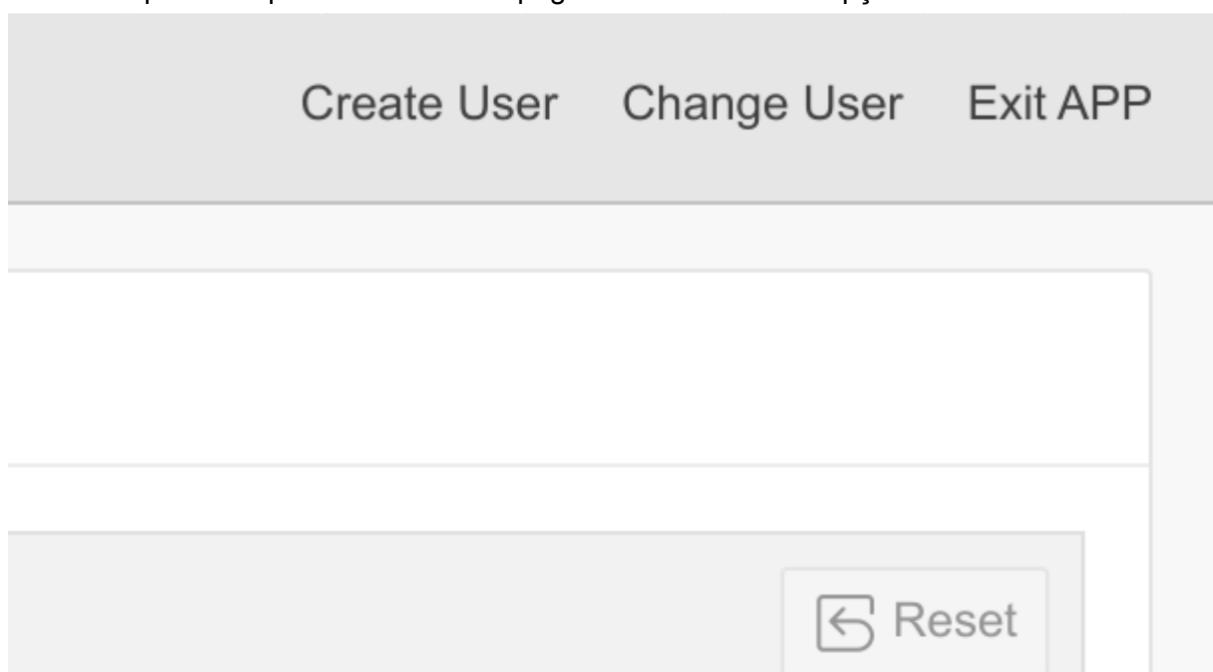
2. Para demitir um desenvolvedor, carregue em 'delete row'



1. Carregue aqui para abrir o menu



10. No lado superior esquerdo de todas as páginas estão diversas opções.



- Para trocar de utilizador carregue em '**Change User**'
- Para criar um novo utilizador carregue em '**Create User**' e siga os passos de 2.b)
- Para sair da aplicação carregue em '**Exit App**'

