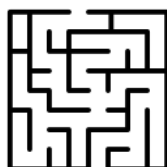

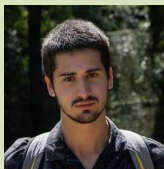
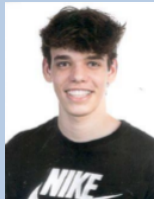
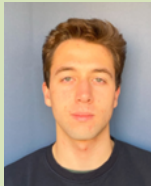










PI de Sistemas de Informação Distribuídos (24/25)

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas



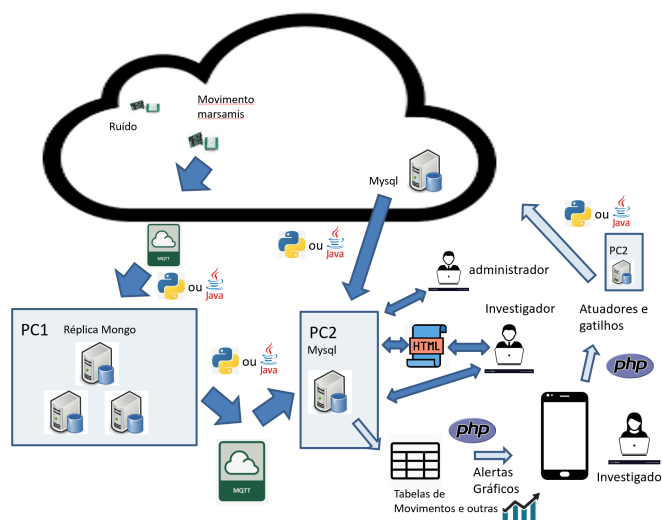
Grupo 34		Grupo 20
<p>111158 Pedro Miguel Vieira pmgav@iscte-iul.pt</p> 		<p>111071 Alexandre Lopushanskyy aggly@iscte-iul.pt</p> 
<p>111357 Tomás Ramalho tabro@iscte-iul.pt</p> 		<p>111567 João Miguel Madureira Louro jmml01@iscte-ul.pt</p> 
<p>111131 Manuel Pina mames1@iscte-iul.pt</p> 		<p>105017 Afonso Padinha apaof@iscte-ul.pt</p> 
<p>106312 Francisco de Sousa Rainho froor2@iscte-iul.pt</p> 		<p>123456 Pedro Nogueira Ramos pnr@iscte-ul.pt</p> 
<p>111358 Francisco Freire Lopes fflse@iscte-iul.pt</p> 		<p>123456 Pedro Nogueira Ramos pnr@iscte-ul.pt</p> 
<p>111393 Tomás Carvalho Silva tcsao1@iscte-iul.pt</p> 		<p>123456 Pedro Nogueira Ramos pnr@iscte-ul.pt</p> 



Instruções

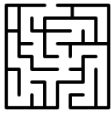
Estas instruções são de cumprimento obrigatório. Relatórios que não cumpram as indicações serão penalizados na nota final.

- Podem (e em várias situações será necessário) ser adicionadas novas páginas ao relatório, mas não podem ser removidas páginas. Se uma secção não for relevante, fica em branco, não pode ser removida;
- A paginação tem de ser sequencial e não ter falhas;
- O índice tem de estar atualizado.
- O grupo que inicia o documento (coluna à esquerda na folha de rosto) preenche apenas a parte inicial (até ao fim da secção 1). Este documento word vai ser colocado no moodle para que o outro grupo (à direita da folha de rosto) possa descarregar e continuar a preenchê-lo (secção 2)

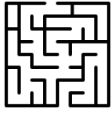


Índice

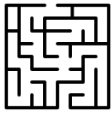
1	Especificação.....	5
1.1	Da Nuvem para o Mongo.....	5



1.2	Descrição Geral do Procedimento de Mongo Para Mysql.....	6
1.4	Tratamento de dados anómalos (valores de sensores errados).....	8
1.5	Tratamento de outliers de temperaturas.....	9
1.6	Tratamento de Alertas de Som.....	10
1.7	Tratamento de número de marsamis numa sala (obter pontuação).....	11
1.8	Especificação de Store Procedures SQL de apoio à migração e tratamento de dados..	12
1.9	Especificação de Triggers de apoio à migração e tratamento de dados.....	13
1.10	Modelo Relacional.....	14
1.11	Utilizadores Base de Dados Mysql.....	15
1.12	Procedimentos Manutenção da Aplicação.....	16
1.13	Eventos de suporte à aplicação (caso existam).....	17
1.14	Consulta por HTML/PHP.....	18
2	Implementação.....	19
	19
2.1	Coleções a criar em cada uma das réplicas do Mongo.....	19
2.2	Descrição Geral do Procedimento de Mongo Para Mysql.....	20
2.3	Tratamento de dados anómalos (valores de sensores errados).....	23
2.4	Tratamento de outliers de temperaturas.....	24
2.5	Tratamento de Alertas de Som.....	25
2.6	Tratamento de número de marsamis numa sala (obter pontuação).....	26
2.7	Implementação de Stored Procedures SQL de apoio à migração e tratamento de dados	27
2.8	Implementação de Triggers.....	28
2.1	Modelo Relacional.....	29
2.2	Utilizadores Base de Dados Mysql.....	30
2.3	Procedimentos Manutenção da Aplicação.....	31
2.4	Eventos de suporte à aplicação (caso existam).....	32
2.5	PrintScreen dos formulários HTML implementados.....	33
2.6	PrintScreen do formulários Android com dados.....	34
	Código de Triggers implementados.....	36



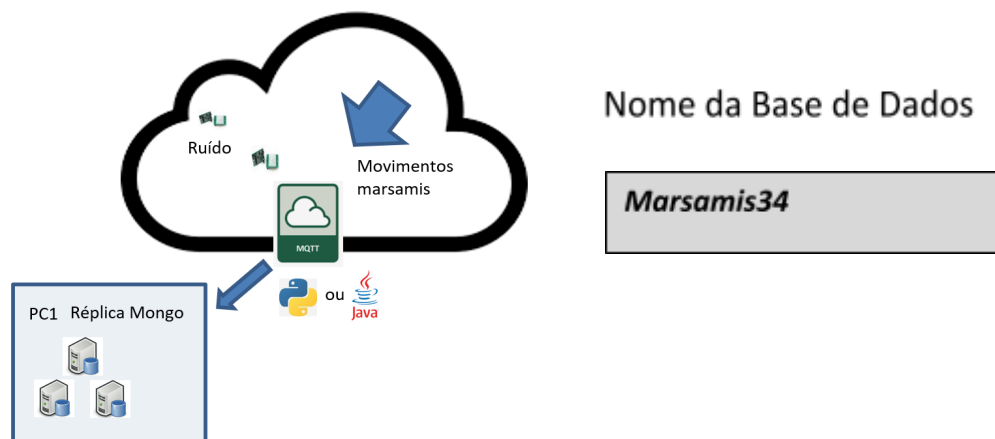
Código Stored Procedures implementados.....	37
---	----



1 Especificação

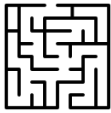
Esta secção é onde o grupo que inicia o documento (coluna à esquerda na folha de rosto) coloca a especificação do que pretende implementar. Mais tarde pode implementar de outra maneira, mas aqui vão as primeiras ideias que serão avaliadas na primeira oral e que vão ser entregues a outro grupo para que analisem e vejam se aproveitam as vossas ideias.

1.1 Da Nuvem para o Mongo



Nome Coleção	O que armazena?
Movimentos	Coleção para armazenar os movimentos dos marsamis entre salas.
Ruído	Coleção para armazenar os valores de ruído recebidos pelos sensores.
Mensagem	Alertas gerados quando o ruído ultrapassa o limite ou quando ocorrem erros.

Para cada coleção exemplifica um documento



Coleção: Movimentos

{

"Player": 1,

"Marsami": 47,

"RoomOrigin": 4

"RoomDestiny": 5,

"Status": 1,

"Hour": Hora a que foi gerada

"migrado": True

}

Nota: **Status pode ser:** 0: Preso, 1: Normal, 2: Cansado

Coleção: Ruído

{

"Player": 1,

"Hour": "2024-07-04 16:29:21.281898",

"Sound": 19.0

"migrado": True

}

Coleção: Mensagem

{

"Player": 2,

"Message": "Ruído ultrapassou o limite",

"Hour": "2024-07-04 16:29:21.281898",

"Level": "Alerta"



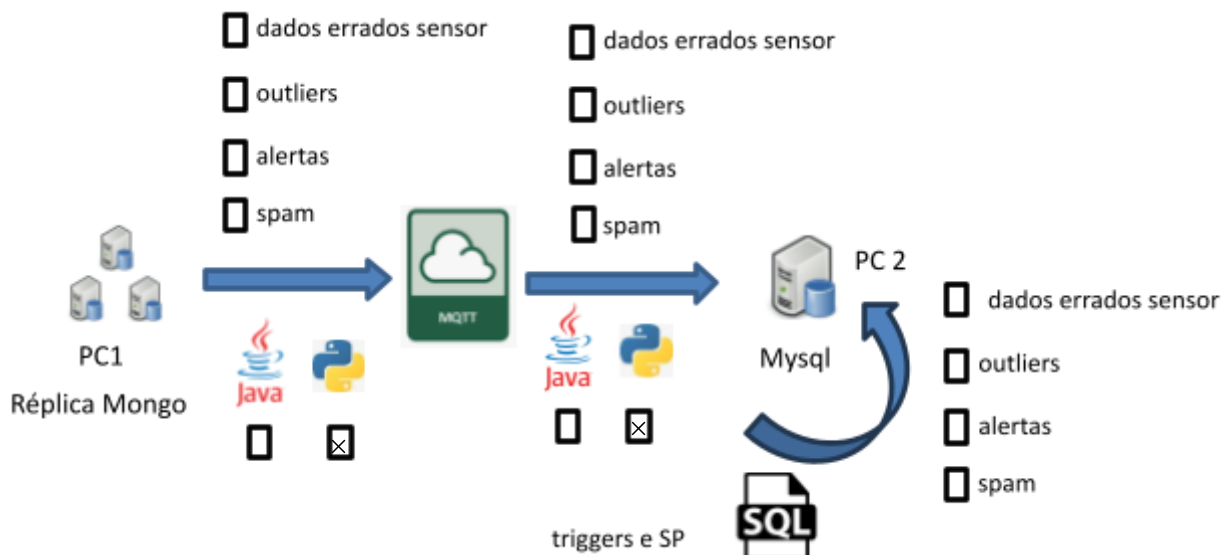
```
"migrado": True
```

```
}
```

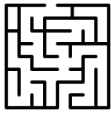
Nota: Level pode ser de 3 tipos: Alerta, Erros, INFO (Ex : marsami 2 cansado)

1.2 Descrição Geral do Procedimento de Mongo Para Mysql

A passagem do Mongo para Mysql tem duas fases: a) enviar de Mongo para MQTT e b) receber de MQTT para Mysql. No diagrama deve ser indicado (nas check box) em qual das fases são programadas (em java e/ou python) as tarefas enumeradas (podem ser na fase a) b) ou ambas).



- Com que “periodicidade” (segundos) o programa vai buscar ao mongo: 2 1 a 5 segundos (3 segundos que é a média)
Esta decisão tem por base o processo de migração que escolhemos, visto que apenas podemos controlar duplicações dos dados (considerando uma comunicação unidirecional), e não perdas de dados na transmissão. Para que não afete a jogabilidade decidimos uma periodicidade menor.
- Como garantem que não enviam duas vezes o mesmo documento do Mongo para o Mysql / MQTT (com base em datas ou booleano ou etc.):



Cada documento numa coleção do **MongoDB** tem um campo especial `_id`, que é gerado automaticamente se não for definido. Esse campo pode ser usado para **rastrear** e garantir que os dados não sejam duplicados ao serem migrados para o **MySQL**.

O campo `_id` é um identificador único, garantindo que cada documento possa ser diferenciado.

No **MySQL**, iremos armazenar esse `_id` na tabela `dados_migrados` para garantir que um mesmo documento **não seja inserido mais de uma vez**.

O campo `id_mongo`, na tabela `dados_migrados`, receberá o valor do `_id` de cada documento do **MongoDB**, utilizaremos a restrição **UNIQUE** para garantir que **não haverá duplicação**.

Agora, sempre que o **MongoDB publicar no MQTT**, o **MySQL** fará a inserção somente se o `_id` ainda não estiver presente na tabela `dados_migrados`.

Para melhorar a performance sugerimos indexar o campo `id_mongo` na tabela `dados_migrados` para tornar as buscas mais rápidas antes da inserção. Uma outra boa prática é adicionar as coleções MongoDB que vão ser migradas, o campo booleano `migrado` que permite que a consulta no mongo seja mais rápida.

- Pensam usar threads do Mongo para MQTT? Sim e do MQTT para Mysql Sim?
- Quantas threads e/ou quantos main em cada um dos passos?:

Fluxo mongo para MQTT:

Thread principal(main): Será necessária apenas uma que será responsável por gerir o ciclo de vida das outras threads, monitorizar os recursos do sistema e responder a eventuais shutdowns.

Threads dedicadas por funcionalidade:

o Thread de leitura/consulta: Monitoriza novas inserções no MongoDB.

o Thread para processamento: Valida, transforma e prepara os dados para publicação.

o Thread que publica no MQTT: Gere a conexão como o broker mqtt e publica mensagens.

Fluxo MQTT para MYSQL:

Threads principais (main): No fluxo de comunicação entre MQTT e MySQL, optamos por utilizar duas threads



distintas para gerenciar dois programas principais. A **Thread 1** é responsável pelo processamento das mensagens relacionadas aos movimentos de marsamis, enquanto a **Thread 2** gerencia as mensagens referentes ao ruído. Essa separação foi implementada para garantir maior independência entre os processos, aumentando a robustez do sistema em caso de falhas. Além disso, essa abordagem permite priorizar o tratamento das mensagens de som, visto que o ruído pode ser um fator crítico, como no caso do encerramento do jogo.

- No transporte MQTT que QoS vão utilizar? Porquê?

Vamos utilizar **QoS 1 (At Least Once)**, pois garante que a mensagem seja entregue pelo menos uma vez, o que é essencial para a **comunicação unidirecional** entre MongoDB e MySQL. QoS 0 não é confiável, e QoS 2 seria mais lento, então escolhemos um equilíbrio entre confiabilidade e desempenho.

Mesmo sendo **unidirecional**, o **QoS 1** no MQTT garante que a mensagem seja **reenviada até o MySQL confirmar a recepção**. Isso funciona porque, no **nível do MQTT**, o broker armazena a mensagem até receber um **ACK (PUBACK)** do cliente subscritor (MySQL).

Ou seja, **não há feedback direto para o MongoDB**, mas o MQTT cuida da garantia de entrega entre o broker e o MySQL.



1.3 Considerações Adicionais sobre o Processo de Migração

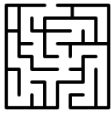
Aqui podem desenvolver informação que considerem relevante relativo ao processo de migração, aspectos que não esteja refletido nas secções seguintes.

Especificação da tabela dados_migrados que irá ser essencial na deteção de duplicados:

```
CREATE TABLE `dados_migrados` (  
  `id` INT(11) AUTO_INCREMENT NOT NULL,  
  `mongo_id` varchar(24) NOT NULL,  
  `data_migração` timestamp DEFAULT  
current_timestamp(),  
  UNIQUE KEY idx_mongo_id_colecao (mongo_id,  
coleção_origem)  
);
```

Criação do índice na tabela dados_migrados:

```
ALTER TABLE dados_migrados ADD UNIQUE INDEX  
idx_id_mongo (id_mongo);
```



1.4 Tratamento de dados anómalos (valores de sensores errados)

Aqui devem explicar o que fazer caso se detectem valores "errados" (datas impossíveis, caracteres estranhos, etc.). Se recorrerem a triggers ou SP então indicam em secção mais adi

ante.

Para garantir a integridade e a confiabilidade dos dados do sistema é essencial criar um mecanismo que trate de dados anómalos. Estes dados podem ser:

- Datas impossíveis
- Caracteres estranhos
- Valores fora do intervalo esperado
- Dados inconsistentes

O tratamento destes dados deve ser feito em dois momentos:

1. Antes de escrever no MongoDB.

O que deve ser validado:

- Formato dos dados
- Intervalos de valores
- Consistência dos dados

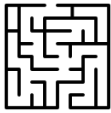
Como implementar:

Antes de escrever os dados no MongoDB, o programa deve validar os campos conforme as regras definidas. Se os dados forem válidos devem ser registados, caso contrário devem ser rejeitados e registados num log.

2. Durante a migração do MongoDB para o MySQL.

O que validar:

- Duplicação de dado
- Consistência dos dados



- Formato dos dados

Como implementar:

- Antes de inserir os dados no MySQL, o programa deve validar os campos conforme as regras definidas.

1.5 Tratamento de outliers de Som

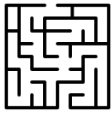
Aquí devem explicar o que fazer caso se detectem "outliers" (valores fisicamente possíveis mas irrealistas, como por exemplo variações muito bruscas do ruído apenas num segundo). Se recorrerem a triggers ou SP então indicam em secção mais adiante.

Para garantir a integridade dos dados de ruído no sistema, é necessário um mecanismo para identificar e tratar outliers. No contexto do jogo, o nível de ruído começa com um valor base e aumenta gradualmente com os movimentos. As variações observadas são pequenas, como por exemplo **0.04 → 0.12 → 0.2 → 0.3**, refletindo a influência dos deslocamentos dentro do labirinto. No entanto, um outlier seria um aumento repentino e irrealista, como um salto de **19.3 para 21.0 (exemplo)** em apenas um segundo, o que não alinha com a dinâmica do sistema.

Caso um outlier seja detectado, a abordagem correta deve ser descartar o valor e manter o último valor confiável.

Para evitar a inserção de valores anómalos na base de dados um **Trigger** no MySQL, denominado por **CheckSoundLevelBeforeInsert()**, que executado antes da inserção de novos valores na tabela sound, verifica a variação do ruído em relação ao último valor registado. Se a diferença for maior que um limite aceitável (por exemplo, **0.5**), a inserção é bloqueada. Isso impede que picos inesperados de ruído sejam armazenados.

Nota: Optamos por 0.5 como limite para considerar um outlier, pois ao analisar os valores *do output do jogo, verificamos que seria uma variação significativa em relação aos valores típicos do conjunto, garantindo que apenas desvios realmente relevantes seriam



identificados, sem capturar flutuações menores que podem ser normais no contexto do jogo.

1.6 Tratamento de Alertas de Som

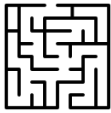
Aqui devem explicar o que fazer caso se detetem situações alarmantes relativas ao som. Se recorrerem a triggers ou SP então indicam em secção mais adiante. Qu situações despoletam os alertas, onde e como são armazenados. Explicar se existem mecanismos para evitar "spam" (demasiadas mensagens). É aconselhável recorrer a esquemas gráficos para explicar o mecanismo.

No caso de não se tratar de outliers, os alertas de som devem ser acionados quando o nível de ruído no labirinto ultrapassa um **limite pré-definido**. Isso pode ser feito com base em dois critérios:

Ruído próximo do valor de referência: Se o nível de ruído detetado está próximo do ruído de referência calculado no início do jogo.

Aumento Súbito do Ruído: Quando há um crescimento muito rápido do nível de ruído em um curto período.

O objetivo é informar o utilizador de alguma forma sobre esses alertas. Para isso, podemos recorrer a um trigger (NotificarRuídoExcessivo) que aciona notificações simples sempre que o nível de ruído ultrapassar um limite pré-definido.



1.7 Tratamento de número de marsamis numa sala (obter pontuação)

Aqui devem explicar como funciona o mecanismo que detecta que o número de marsamis odd é (ou vai ser) igual ao número de marsamis even. Como detecta, e o que desencadeia.

Para garantir a correta deteção do equilíbrio, na tabela OcupacaoLabirinto, que mantém a contagem de Marsamis Odd e Even por sala. Sempre que um Marsami entra ou sai, um trigger (**MarcarEquilibrioOddEven**) verifica se o número de Odd é igual ao de Even e armazena um estado booleano (equilibrio_atingido) na sala correspondente. Se o número de marsamis odd for igual ao de evens o campo equilibrio_atingido na tabela Sala é atualizado para TRUE e vice-versa.

1.8 Especificação de Store Procedures SQL de apoio à migração e tratamento de dados

Nas secções anteriores foram descritos mecanismos que podem ou não necessitar de recorrer a Store Procedures. É nesta tabela que eles deverão ser listados. Na descrição apenas colocar informação que não seja óbvia.

RegistarMovimento	Marsami; sala_origem; sala_destino;	Regista o movimento de marsamis entre salas, atualizando a tabelas ocupacaolabirinto e medicoespassagem
RegistarRuido	valor_ruido	Regista o valor de ruido atual do labirinto.
AtualizarEstadoMarsami	Marsami; Status;	Atualiza o estado de um Marsami (Ex: 0-Preso, 1-normal, 2-Cansado) na tabela medicoespassagem no campo Status
ConsultarOcupacaoPorSala	ID_SALA (nº da sala que	Tem como objetivo consultar a

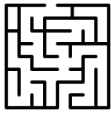


	queremos consultar)	ocupação de uma sala específica.
PermirGatilho	ID_JOGO ID_SALA	Verifica se a sala tem equilíbrio entre odds e evens através do campo <code>equilibrio_atingido</code> na tabela <code>Sala</code> . Se sim, dá +1 ponto ao jogador. Se não, penaliza com -0.5 pontos. Finalizar jogo em caso de <code>pontuação<=0</code> .

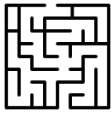
1.9 Especificação de Triggers de apoio à migração e tratamento de dados

Nas secções anteriores foram descritos mecanismos que podem ou não necessitar de recorrer a Triggers. É nesta tabela que eles deverão ser listados. Nas Notas apenas colocar informação que não seja óbvia.

AtualizarOcupacaoSala	OcupacaoLabirinto	I	After	Atualizar a ocupação das salas quando um marsami se move. Especificando odds e evens, fazendo as respetivas contagens.
MarcarequilibrioOddEven	OcupacaoLabirinto	U	After	Se o número de marsamis Odd for igual ao de evens, o campo <code>equilibrio_atingido</code> na tabela <code>Sala</code> é atualizado para <code>TRUE</code>



NotificarRuídoExcessivo	Sound	I	After	Se o valor de ruído que foi inserido estiver a 20% do limite definido.
NotificarEstadoJogo	Jogo	U	After	Notifica o jogador em que estado o jogo se encontra após este ser alterado. Estados possíveis do jogo: A decorrer, Terminado.
NotificarPontuação	Jogo	U	After	Gerar mensagem para notificar o jogador acerca da sua pontuação atual e se ganhou ou perdeu pontos após acionar o gatilho.
VerificarLimiteGatilhos	Sala	U	Before	Verifica se o número de gatilhos acionados numa sala já atingiu o limite de 3. Se sim, impede novas ativações e notifica o jogador que chegou ao limite.
CheckSoundLevelBeforeInsert	Sound	I	Before	Se a variação do ruído for superior a 0.5 em relação ao último valor registado, a inserção é rejeitada para evitar outliers.





1.10 Modelo Relacional

Diagrama relacional completo. Alterações à base de dados original terão de ser justificadas aqui. Caso seja pertinente podem ser adicionadas comentários a justificar opções pouco óbvias.

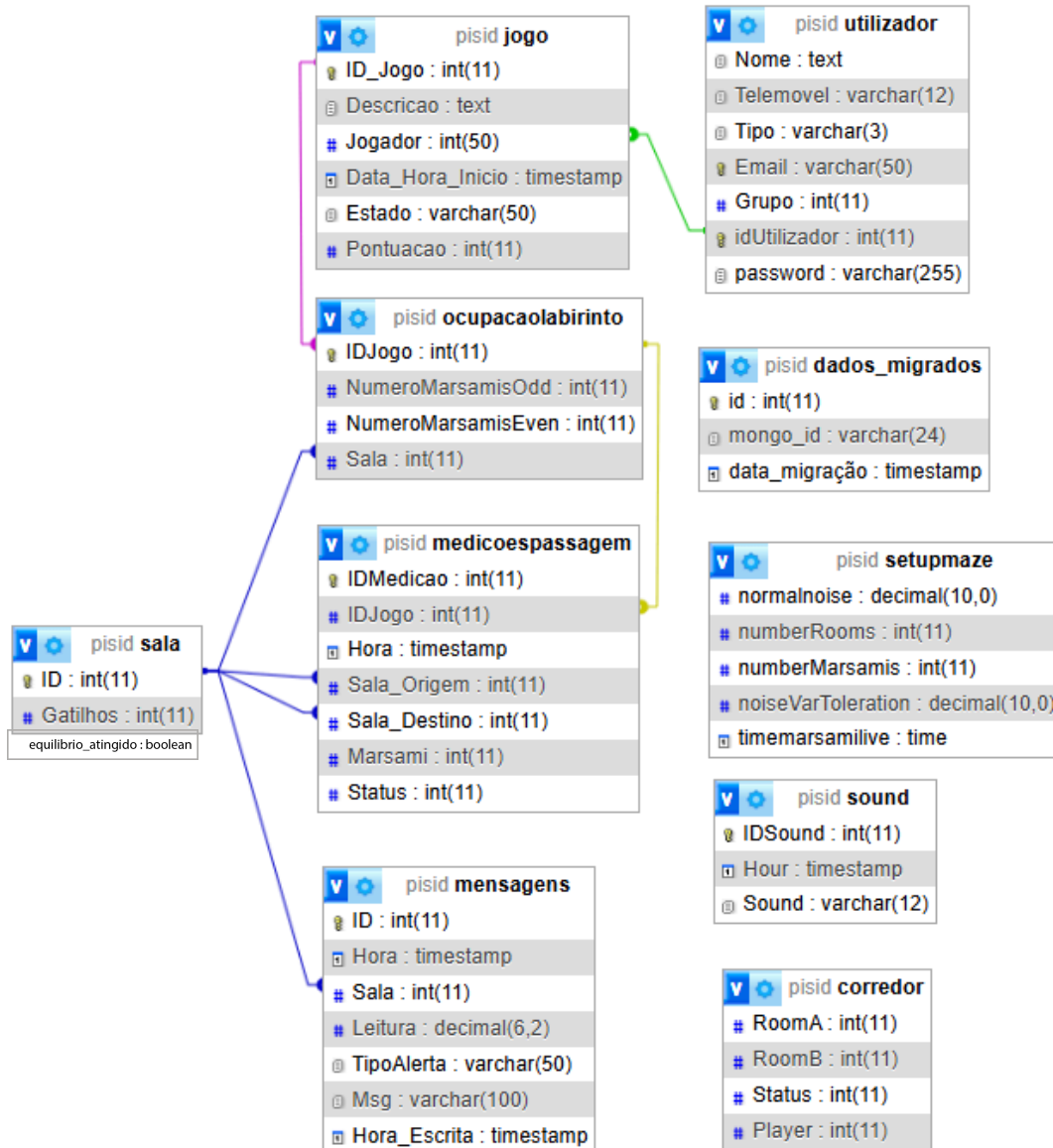
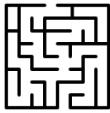
Alterações da base de dados original:

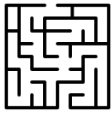
- **Criação da Tabela "Sala":** Tem como chave primária um atributo do tipo inteiro designado "ID" e um atributo designado "Gatilhos" que representa o número de gatilhos que foram acionados na respetiva sala. Por termos criado esta tabela, levou a que todos os atributos que representem a sala nas outras tabelas sejam chaves estrangeiras que referenciam a chave primária ("ID") da tabela Sala. Para além disso adicionamos um atributo do tipo booleano designado "equilíbrio_atingido" que indica se o equilíbrio entre o número de marsamis odd e even foi atingido.
- **Criação da Tabela "dados_migrados":** Tem como chave primária um atributo do tipo inteiro designado "id", um atributo do tipo varchar designado "mongo_id" que irá ser útil para verificar se os dados foram duplicados no transporte do mongoDB para o MySQL. Para além disso temos um atributo do tipo timestamp designado "data_migração" que representa o exato instante que foi realizada a respetiva migração.
- **Alteração da Tabela "Utilizador":** Foi adicionado um atributo do tipo inteiro designado "idUtilizador" que representa a chave primária da tabela Utilizador. Este atributo serve para identificar o respetivo jogador, sendo que o atributo "Jogador" da tabela "jogo" vai ser a chave estrangeira que referencia este atributo. Além disso, foi adicionado um atributo "password" do tipo VARCHAR(255) (tamanho que garante espaço suficiente para formato hash), que armazena a password do utilizador de forma segura. Esse campo é essencial para a autenticação dos utilizadores no sistema. É então necessário atualizar os scripts responsáveis pela leitura dessa tabela no Android. Isso significa que os ficheiros e funções que fazem queries na tabela



utilizador devem ser modificados para incluir o novo campo password, garantindo que os dados sejam corretamente manipulados pela aplicação.

- **Alteração da Tabela "Jogo":** Foi adicionado um atributo do tipo inteiro designado "Pontuacao" que tem como objetivo guardar a pontuação do jogador num respetivo jogo.





1.11 Utilizadores Base de Dados Mysql

Nesta secção deverá ser explicado de que forma deverá ser feita a manutenção de utilizadores Mysql. Nomeadamente deverá ser indicado, para cada tipo de utilizador, que privilégios ele tem sobre que tabelas e Stored Procedures (todos os SP usados na Aplicação

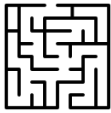
De acordo com a nossa especificação vão existir três tipos de utilizadores: O administrador da aplicação, o administrador da base de dados e o jogador.

O administrador da aplicação gere os jogos, os jogadores e pode reinicializar a migração de dados do mongoDB para o mySQL caso ocorram falhas no processo automático das migrações das leituras dos sensores.

O administrador da base de dados é responsável pela gestão da base de dados incluindo a criação e remoção de tabelas, Stored Procedures e gestão de utilizadores.

O jogador gere os seus jogos e interage com o labirinto (abrir/fechar portas, acionar gatilhos, visualizar estado das salas e dos marsamis).

	Administrador da Aplicação	Administrador da Base de dados	Jogador
Utilizador	U/I/D/L	U/I/D/L	-
Dados_Migrações	U/I/D/L	U/I/D/L	-
Sound	L	U/I/D/L	L
OcupacaoLabirinto	L	U/I/D/L	L
SetupMaze	U/I/D/L	U/I/D/L	-
Sala	L	U/I/D/L	-
MedicoesPassagem	L	U/I/D/L	L



Jogo	U/I/D/L	U/I/D/L	U/I/L
Mensagens	L	U/I/D/L	L
Corredor	U/I/D/L	U/I/D/L	L
Criar_utilizador	X	X	-
Remover_utilizador	X	X	-
Editar_utilizador	x	x	x
Criar_jogo	X	X	X
Apagar_jogo	-	X	x
Editar_Jogo	-	x	x
Selecionar_Jogo	-	x	x
Iniciar_Jogo	-	x	x
Login	x	x	x
RegistarMovimento	-	x	-
RegistarRuido	-	x	-
AtualizarEstadoMarsa mi	-	x	-
ConsultarOcupacaoPorSala	-	x	x
PermirGatilho	-	x	x

Em que U=Update, I Insert, D- Delete, L=Leitura, X=Executar SP e - sem permissões.



1.12 Procedimentos Manutenção da Aplicação

Nesta secção deverão ser listados os SP para a manutenção de utilizadores e jogos (apenas os obrigatórios)

Login	Email, Password	Faz Login com a conta de utilizador
Criar_utilizador	Nome, Telemovel, Tipo, Email, Grupo	Cria um utilizador na aplicação
Remover_utilizador	Id_Utilizador	Remove um utilizador com base no id
Editar_utilizador	Nome, Telemovel, Password, Email, Grupo	Editar dados do utilizador
Criar_jogo	Descricao, Jogador, Data_Hora_Inicio, Estado	Regista um novo jogo na base de dados
Apagar jogo	ID_Jogo	Remove um jogo
Selecionar_Jogo	ID_Jogo	Seleciona um jogo existente para jogar
Editar_jogo	Descricao, Jogador, Data_Hora_Inicio, Estado	Edita dados de um jogo existente
Iniciar_Jogo	ID_Jogo	Dá início ao jogo criado



1.13 Eventos de suporte à aplicação (caso existam)

Deverão ser indicados todos os eventos relevantes para o processo de migração, eventos do Windows e do Mysql. Por eventos entende-se as tarefas do Windows ou eventos do Mysql

Backup Automático da Base de Dados	Windows/MySQL	Realiza backups periódicos da base de dados para evitar perda de informações.
Limpeza de Dados Antigos	MySQL	Remove registos desnecessários ou muito antigos para otimizar espaço e desempenho.



1.14 Consulta por HTML/PHP

Desenhar o layout dos formulários pretendidos, se relevante colocar texto a explicar a funcionalidade pretendida. Formulários:

- *Fazer login*
- *Criar (ou seleccionar de uma lista de jogos um para alterar) um jogo) e, para esse jogo, editar os valores associados. Quando está a alterar não pode alterar valores de chaves estrangeiras e primárias.*

Indicar para cada botão qual o SP que deverá ser executado

Menu Inicial

Editar Utilizador

Login

Criar Jogo

Seleccionar Jogo



Editar Utilizador

NOME	<input type="text" value="jogador1"/>
EMAIL	<input type="text" value="jogador1@gmail.com"/>
GRUPO	<input type="text" value="v"/>
TELEMOVEL	<input type="text" value="914678925"/>
PASSWORD	<input type="password" value="*****"/>

Guardar

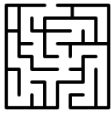
Botão "Guardar" -> SP_Editar_utilizador

Restrições: Password- Deve ter um formato válido (exemplo: mínimo de 8 caracteres, contendo letras maiúsculas, minúsculas, números e símbolos).

Nome- Não pode ser um nome que já está em uso.

Email-Deve seguir o padrão normal de email, exemplo válido: exemplo@email.com . Para além disso este não pode ser um email duplicado.

Telemóvel- Apenas números. Não são permitidas letras.
Exemplo valido, (916357405)



Login

EMAIL

jogador1@gmail.com

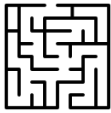
PASSWORD

login

Botão "login" -> SP_Login

Email - No caso de o email estar errado enviar uma mensagem, a dizer email incorreto.

Password- No caso da password estar errada enviar uma mensagem, a dizer password incorreta.



Lista de jogos

Jogo1

Jogo2

Jogo3

Jogo4

Jogo5

Voltar

Botão "JogoX" -> SP_Selecionar_Jogo

Jogo X

Editar Jogo

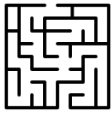
Iniciar Jogo

Remover Jogo

Voltar

Botão Iniciar Jogo->SP_Iniciar_Jogo

Botão Remover Jogo->Apagar_jogo



Editar Jogo

DESCRICAO

JOGADOR

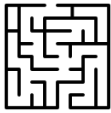
Guardar Jogo

Botão "Guardar Jogo" -> SP_Editar_Jogo

Restrições:

Descrição- Deve ter tamanho mínimo de 20 caracteres, pode conter letras, números e caracteres especiais básicos.

Jogador-idUtilizador tem de estar registado na Base de Dados senão lança mensagem de erro.



Criar Jogo

DESCRICAO

JOGADOR

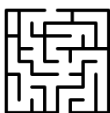
Criar Jogo

Botão "Criar Jogo" -> SP_Criar_jogo

Restrições:

Descrição- Deve ter tamanho mínimo 20 caracteres, pode conter letras, números e caracteres especiais básicos.

Jogador-idUtilizador , tem de estar registado na Base de Dados senão lança mensagem de erro.



2 Implementação

*Esta secção é para ser preenchida pelo grupo que recebeu o documento. Aqui vão falar da implementação que fizeram. A implementação é o "best of", ou seja, o que **agora** acham que é a melhor solução, com base em tudo o que aprenderam (com as vossas experiências, com as ideias do outro grupo, discussões com o professor, google, chatgpt, etc), no limite podem não seguir nada do que tinham especificado no documento que entregaram ao outro grupo.*

2.1 Coleções a criar em cada uma das réplicas do Mongo

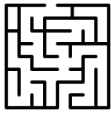
Versão	Número Coleções
Especificação inicial	2
Recebida outro grupo	3
<u>Implementada</u>	2

Justificação da escolha

Texto justificativo da opção final. Se for idêntica à inicial (enviada a outro grupo) não preencher

Para cada coleção **implementada** exemplifica um documento

Coleção : movimentos



```
{  
  "_id": {  
    "$oid": "6814b9edd461cefd6bddfa23"  
  },  
  "Player": 20,  
  "Marsami": 8,  
  "RoomOrigin": 2,  
  "RoomDestiny": 5,  
  "Status": 1,  
  "enviado": true  
}
```

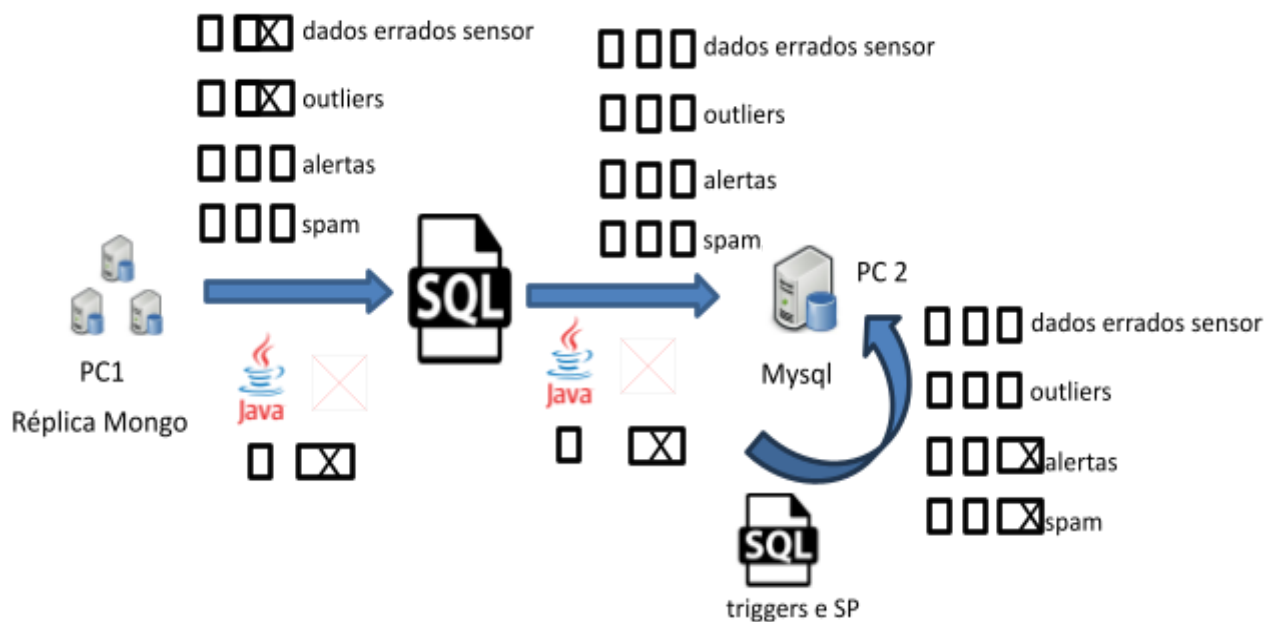
Coleção : nivel som

```
{  
  "_id": {  
    "$oid": "6814b9edd461cefd6bddfa24"  
  },  
  "Player": 20,  
  "Hour": "2025-05-02 13:26:23.740274",  
  "Sound": 19.56,  
  "enviado": true  
}
```




2.2 Descrição Geral do Procedimento de Mongo Para Mysql

Na checkbox da esquerda indicam o que especificaram inicialmente, na do meio a especificação do outro grupo e na da direita a vossa implementação.



Justificação da escolha

Tratámos dos alertas e spam através de triggers, visto que os dados estão a ser lidos através das tabelas do mysql, para posteriormente enviar mensagens de alerta.

Os dados errados de sensor e outliers optámos por fazer a procura logo na passagem de mqtt para mongodb, de modo a não preencher o mongodb com dados incorretos.



Versão	Periodicidade vai buscar Mongo	Como evitam enviar duas vezes para MQTT	Número Threads	QOS
Especificação inicial	valor do delay - 1s	boolean “enviado”	1	1
Recebida outro grupo	1 a 5 segundos (3 segundos que é a média)	id de cada mensagem no Mongoddb	2	1
<u>Implementada</u>	1 segundo	boolean “enviado”	1	1

Justificação da escolha

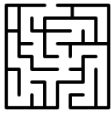
A periodicidade escolhida é 1 segundo porque este valor de tempo estará abaixo da periodicidade enviada pelo jogo.

Usamos o boolean por acharmos ser a solução mais robusta e por ter sido uma solução aprovada pelo professor.

Achamos que 1 thread é suficiente para a tarefa que estamos a fazer, visto que pelos testes que fizemos não se perderam mensagens na passagem de mongo para mqtt.

QOS 1 pois garantimos que as mensagens são recebidas, contudo também podemos receber valores duplicados, mas tratamos deste problema através do campo boolean.

Usamos QOS 2 apenas na passagem de mqtt para mysql.

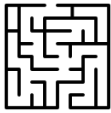


Nas próximas quatro secções devem, na descrição, resumidamente descrever num texto escorreito e legível, a forma como foi implementada. Têm de ficar muito explicitamente indicado a o que não resultou da especificação inicial (cor preta), o que foi aproveitado da especificação que receberam de outro grupo (cor azul) e o que resultou de ideias posteriores vossas (cor verde). Na justificação sejam muito objectivos a explicar a razão de terem alterado a especificação inicial

Para que os dados fossem corretamente interpretados e armazenados, tivemos de passar as mensagens sem nada e as mensagens com plicas(") a mensagens com aspas(“”).

- Datas impossíveis
- Valores Fora do intervalo esperado
- Dados inconsistentes
- Formato dos dados
- Verificação se as mensagens de movimento são de corredores conectados(ligação à bd nuvem).
- Dados são apagados e não enviados para o mongodb.

A nossa especificação estava incompleta em relação ao tratamento de dados anómalos, então decidimos que era necessário acrescentar algumas ideias da especificação do grupo 34 e outras ideias novas.



2.3 Tratamento de dados anómalos (valores de sensores errados)

- Valores de som negativos ou acima de 23 considerámos dados anómalos (sensor errado)

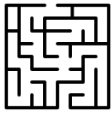
2.4 Tratamento de outliers de temperaturas

1 Descrição

2 Justificação das alterações caso tenham havido

2.5 Tratamento de Alertas de Som

Usamos um trigger em MySQL “alertaRuido”. Sempre que uma linha é inserida na tabela “sound”, o trigger verifica se o valor do som é superior a



21. Caso seja, significa que se está a aproximar do valor limite (21,5). Insere uma linha na tabela “mensagens” com este alerta e o script python que envia as mensagens para o tópico “pisid_mazeact” fica à espera de uma mensagem de alerta. Quando deteta, envia a mensagem MQTT “CloseAllDoor” de modo a que o ruído do labirinto diminua.

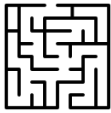
2.6 Tratamento de número de marsamis numa sala (obter pontuação)

Usamos um trigger em MySQL “trigger_ativar_gatilho”. Para cada update da tabela “ocupacaolabirinto” deteta se NumeroMarsamisOdd = NumeroMarsamisEven. Se sim, insere uma linha da tabela mensagens com esse alerta, indicando a também a sala correspondente. O script python dos atuadores aguarda que receba uma linha de alerta com o campo Msg=“ativar gatilho” e envia a mensagem mqtt “Score” para o tópico “pisid_mazeact”, com a sala especificada.

2.7 Implementação de Stored Procedures SQL de apoio à migração e tratamento de dados

É nesta tabela que deverão ser listados os SP que implementam mecanismos anteriores. Na quarta coluna têm de colocar um dos seguintes símbolos:

= - igual ao especificado
A(og) – Alterado com base em ideia de outro grupo
A(ni) – Alterado com base em novas ideias
N(og) - Novo com base em ideia de outro grupo
N(ni) – Novo com base em ideias novas
A – Alterado com base em novas ideias

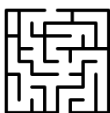


...	

*Não Implementámos SP's de apoio à migração e tratamento de dados.
Essa parte está nos scripts python/triggers.*

É nesta tabela que deverão ser listados os triggers Na sexta coluna têm de colocar um dos seguintes símbolos:

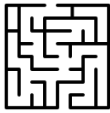
= - igual ao especificado
A(og) – Alterado com base em ideia de outro grupo
A(ni) – Alterado com base em novas ideias
N(og) - Novo com base em ideia de outro grupo
N(ni) – Novo com base em ideias novas
A – Alterado com base em novas ideias



2.8 Implementação de Triggers

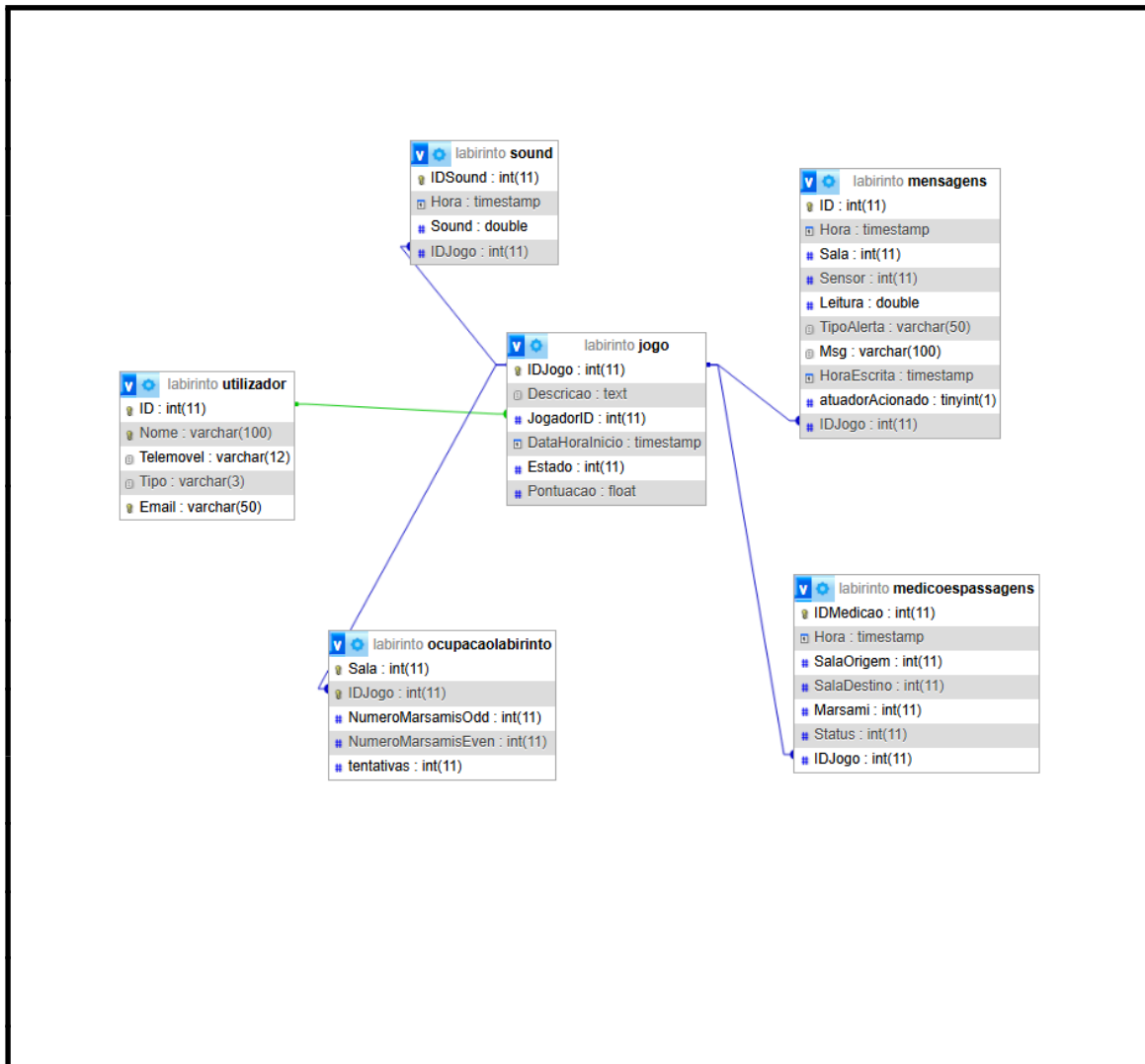
Nome Trigger	Tabela	Tipo de Operação (I,U,D)	(After, Before)	Notas	
alertaRuido	sound	I	After	Alerta enviado para tabela mensagens sempre que o valor de som se estiver a aproximar do limite	=
atualizar_ocupacao_labirinto	medicoe spassagens	I	After	atualiza a tabela ocupacaolabirinto com base nas mensagens de movimento recebidas	N(ni)
trigger_ativar_gatilho	ocupacaolabirinto	U	After	À medida que distribuição dos marsamis por salas está a acontecer, o trigger deteta se uma sala tem marsamisOdd=marsamisEven e envia o alerta para a tabela mensagens	=
cria_salas_jogo	jogo	I	After	Cria as salas na tabela ocupacao labirinto, sempre que é criado um novo jogo.	N(i)

Usámos esses 4 triggers, porque foram os que achámos necessários para tratar a migração dos dados e ir alterando as informações na base de dados em tempo real.

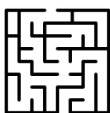


1.1 Modelo Relacional

Diagrama relacional completo implementado. Assinalar a azul alterações derivadas de outro grupo e a verde alterações novas.



Decidimos que o IDJogo necessita estar em quase todas as tabelas, de modo que se saiba a que jogo pertencem as informações. Adicionámos também alguns campos necessários como: "tentativas" na tabela ocupacaolabirinto para registar quantas vezes o gatilho foi enviado para cada sala; atuadorAcionado na tabela mensagens para marcar que aquela mensagem de alerta teve a respetiva mensagem MQTT enviada.



1.2 Utilizadores Base de Dados Mysql

Nesta secção deverão ser indicados os utilizadores e perfis implementados (têm de consta todos os SP usados). Assinalar a azul alterações derivadas de outro grupo e a verde alterações novas.

Tabela	Tipo de Utilizador		
	Administrador	Jogador	...
jogo	U/I/D/L	U, I, L	
utilizador	U/I/D/L	U, L	
ocupacaola birinto	U/I/D/L	L	
medicoespa ssagens	U/I/D/L	L	
mensagens	U/I/D/L	-	
sound	U/I/D/L	L	
Stored Proc.			
CriarUtili zador	X	-	
AlterarDes cricaoJogo	X	X	
CriarJogo	X	X	
EditarUtil izador	X	X	
CriarJogoA dmin	x	-	

Em que U=Update, I Insert, D- Delete, L=Leitura, X=Executar SP e - sem permissões.

Achámos apenas necessários existirem 2 tipos de utilizadores da bd. O administrador que controla tudo e o Jogador que apenas tem acesso a alguns SPs e tabelas, em linhas que tenham a sua informação.

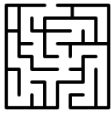


1.3 Procedimentos Manutenção da Aplicação

É nesta tabela que deverão ser listados os SP que implementam mecanismos anteriores. Na quarta coluna têm de colocar um dos seguintes símbolos:

= - igual ao especificado
A(og) – Alterado com base em ideia de outro grupo
A(ni) – Alterado com base em novas ideias
N(og) - Novo com base em ideia de outro grupo
N(ni) – Novo com base em ideia novas
A – Alterado com base em novas ideias

Nome SP	Argumentos	Descrição	
CriarUtilizador	p_nome,p_password	Cria um utilizador (jogador) na bd e escreve a informação na tabela utilizador(nome). Além disso, dá acesso aos SP's que são possíveis de executar por um jogador. Cria também uma view para o jogador poder visualizar os seus jogos.	N(ni)
AlterarDescricaoJogo	p_idJogo,p_novaDescricao	Altera a descrição de um jogo escolhido. Apenas se o jogo for do utilizador que está autenticado no momento, na base de dados.	N(ni)
CriarJogo	p_descricao	Cria dados de um jogo novo na tabela jogo	A(og)
CriarJogoAdmin	p_JogadorID,p_Descricao,p_Estado	Permite ao administrador criar um jogo com o id de um utilizador existente.	N(ni)



EditarUtiliz ador	p_telemovei,p_em aíl	Altera os dados do utilizador atual que estão nos argumentos.	A(og)
----------------------	-------------------------	--	-------

1.4 Eventos de suporte à aplicação (caso existam)

É nesta tabela que eles deverão ser listados todos os eventos relevantes para o processo de migração Na quarta coluna têm de colocar um dos seguintes símbolos:

= - igual ao especificado
A(og) – Alterado com base em ideia de outro grupo
A(ni) – Alterado com base em novas ideias
N(og) - Novo com base em ideia de outro grupo
N(ni) – Novo com base em ideia novas
A – Alterado com base em novas ideias

Texto justificativo da opção final. O que for idêntico à inicial (enviada a outro grupo) não preencher



1.5 PrintScreen dos formulários HTML implementados

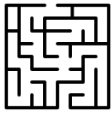
Login

Nome de utilizador:

Password:

Entrar

Ecrã inicial, login



PI de Sistemas de Informação Distribuídos, 23/24

Login

Nome de utilizador:

Jose

Password:

...

Entrar

Ecrã inicial, fazer login do utilizador “Jose”, com a palavra passe “123”

Painel de Utilizador: Jose

Criar Jogo

Editar Utilizador

Alterar Descrição Jogo

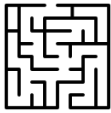
Seus Jogos

ID	Descrição	Estado	Pontuação
----	-----------	--------	-----------

Seus Dados

Nome	Email	Telemóvel	Tipo
Jose	n/a	n/a	U

Ecrã após fazer o login, com os botões criar jogo, editar utilizador, e alterar descrição do jogo



Painel de Utilizador: Jose

[Criar Jogo](#)

[Editar Utilizador](#)

[Alterar Descrição Jogo](#)

Seus Jogos

ID	Descrição	Estado	Pontuação
129	n/a	1	0

Seus Dados

Nome	Email	Telemóvel	Tipo
Jose	n/a	n/a	U

Ecrã após criar jogo, cria um jogo e mostra a tabela desse jogo

Painel de Utilizador: Jose

[Criar Jogo](#)

[Editar Utilizador](#)

[Alterar Descrição Jogo](#)

Editar Utilizador

Novo email:

Novo telemóvel:

[Salvar Alterações](#)

Seus Jogos

ID	Descrição	Estado	Pontuação
129	n/a	1	0

Seus Dados

Nome	Email	Telemóvel	Tipo
Jose	n/a	n/a	U

Ecrã após clicar no botão para editar utilizador, abre dois espaços um para adicionar o email e outro para adicionar o número de telemóvel



Painel de Utilizador: Jose

[Criar Jogo](#)

[Editar Utilizador](#)

[Alterar Descrição Jogo](#)

Seus Jogos

ID	Descrição	Estado	Pontuação
129	n/a	1	0

Seus Dados

Nome	Email	Telemóvel	Tipo
Jose	jose@iscte-iul.pt	960000000	U

Ecrã após editar utilizador, a tabela referente aos dados do utilizador é atualizada

Painel de Utilizador: Jose

[Criar Jogo](#)

[Editar Utilizador](#)

[Alterar Descrição Jogo](#)

Alterar Descrição Jogo

ID do jogo:

129

Nova descrição:

jogo teste jose

[Alterar Descrição](#)

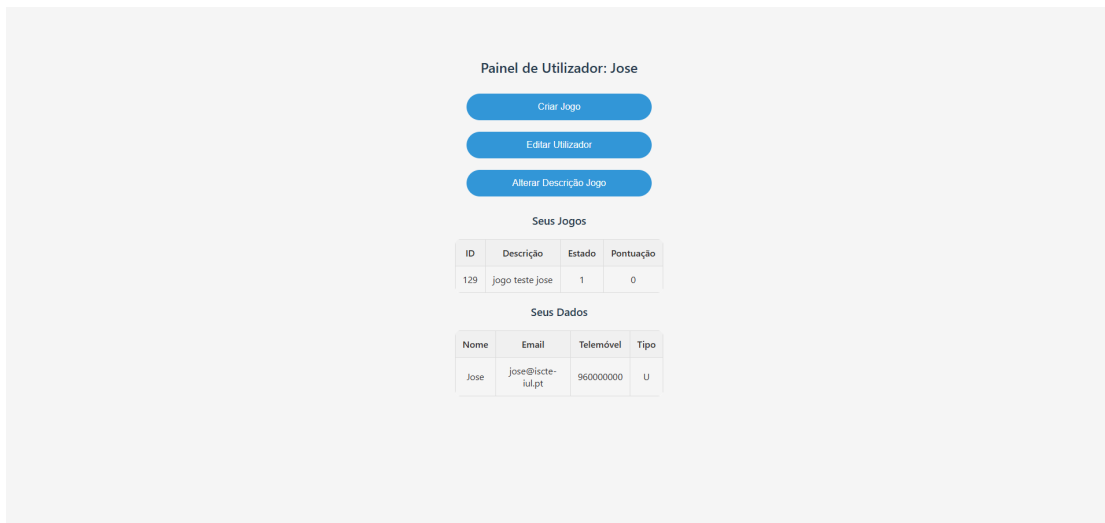
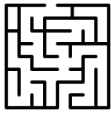
Seus Jogos

ID	Descrição	Estado	Pontuação
129	n/a	1	0

Seus Dados

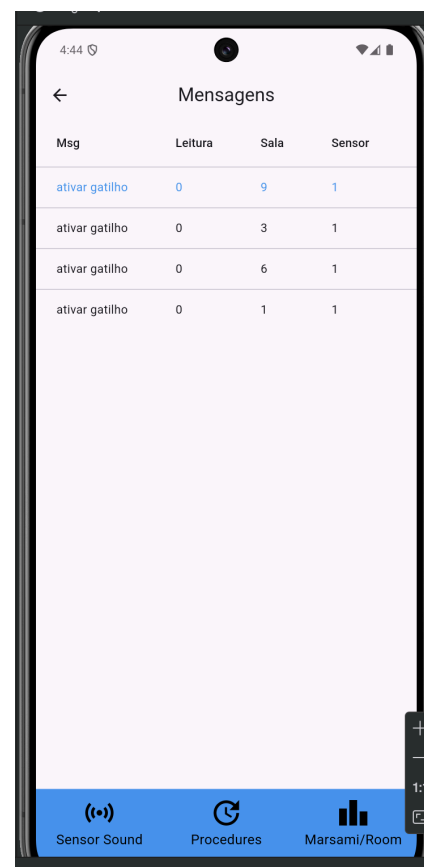
Nome	Email	Telemóvel	Tipo
Jose	jose@iscte-iul.pt	960000000	U


Ecrã após clicar no botão para editar descrição do jogo, abre dois espaços para introduzir o id do jogo onde vamos querer editar a sua descrição



Ecrã após editar descrição do jogo editar, a tabela referente ao jogo com o id passado é atualizada com a descrição dada e ecrã final após testar todos os botões

1.6 PrintScreen do formulários Android com dados





9:31

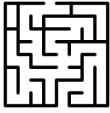
Mensagens

Msg	Leitura	Sala
fechar portas	21.36666666666667	
fechar portas	21.2	
fechar portas	21.04798023816664	
fechar portas	21.215053374894836	
ativar gatilho	0	3
ativar gatilho	0	1
ativar gatilho	0	7
ativar gatilho	0	10
ativar gatilho	0	1
ativar gatilho	0	4

+

1:1

Sensor Sound Procedures Marsami/Room



Anexo Código SQL

Código de Triggers implementados

```
1. Nome Trigger: _alertaRuido
// Alerta de som que envia linha para tabela mensagens

BEGIN
    -- Se o valor de Sound for maior que 21, insere na
    tabela mensagens
    IF NEW.Sound > 21 THEN
        INSERT INTO mensagens (
            Hora,
            Sala,
            Sensor,
            Leitura,
            TipoAlerta,
            Msg,
            HoraEscrita,
            atuadorAcionado,
            IDJogo
        ) VALUES (
            NEW.Hora,
            NULL,
            2,
            NEW.Sound,
            'sound',
            'fechar portas',
            NOW(),
            0,
            NEW.IDJogo
        );
    END IF;
END
```



```
2. Nome Trigger: _atualizar_ocupacao_labirinto____
// Vai atualizando a tabela ocupacaolabirinto à medida
que recebe linhas da tabela medicoespassagens

BEGIN
    DECLARE origem INT;
    DECLARE destino INT;
    DECLARE marsami INT;
    DECLARE jogo_id INT;

    SET origem = NEW.SalaOrigem;
    SET destino = NEW.SalaDestino;
    SET marsami = NEW.Marsami;
    SET jogo_id = NEW.IDJogo;

    -- Atualiza a sala de origem (decrementa Marsami)
    IF origem <> 0 THEN
        IF MOD(marsami, 2) = 1 THEN
            -- Marsami ímpar
            UPDATE ocupacaolabirinto
            SET NumeroMarsamisOdd =
GREATEST(NumeroMarsamisOdd - 1, 0)
            WHERE Sala = origem AND IDJogo = jogo_id;
        ELSE
            -- Marsami par
            UPDATE ocupacaolabirinto
            SET NumeroMarsamisEven =
GREATEST(NumeroMarsamisEven - 1, 0)
            WHERE Sala = origem AND IDJogo = jogo_id;
        END IF;
    END IF;

    -- Atualiza a sala de destino (incrementa Marsami)
    IF destino <> 0 THEN
        IF MOD(marsami, 2) = 1 THEN
            -- Marsami ímpar
            UPDATE ocupacaolabirinto
            SET NumeroMarsamisOdd = NumeroMarsamisOdd + 1
            WHERE Sala = destino AND IDJogo = jogo_id;
        ELSE
            -- Marsami par
            UPDATE ocupacaolabirinto
            SET NumeroMarsamisEven = NumeroMarsamisEven +
```

1



```
        WHERE Sala = destino AND IDJogo = jogo_id;
    END IF;
END IF;

END

3. Nome Trigger: __trigger_ativar_gatilho__
// a cada update da tabela ocupacao labirinto verifica se
marsamisodd=marsamiseven, se sim envia alerta para a
tabela mensagens.

BEGIN
    DECLARE num_tentativas INT;
    DECLARE ultima_hora DATETIME;

    -- Verifica número de tentativas atuais na sala
    SELECT tentativas INTO num_tentativas
    FROM ocupacaolabirinto
    WHERE Sala = NEW.Sala AND IDJogo = NEW.IDJogo;

    -- Obtém a hora do último alerta de gatilho (se
    existir)
    SELECT MAX(Hora) INTO ultima_hora
    FROM mensagens
    WHERE Msg = 'ativar gatilho'
    AND Sala = NEW.Sala
    AND IDJogo = NEW.IDJogo;

    -- Verifica se passou pelo menos 5 segundos desde o
    último alerta (evitar spam)
    IF NEW.numeromarsamisodd = NEW.numeromarsamiseven
    AND NEW.numeromarsamisodd > 0
    AND num_tentativas < 3
    AND (ultima_hora IS NULL OR TIMESTAMPDIFF(SECOND,
    ultima_hora, NOW()) >= 5) THEN

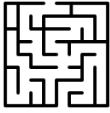
        INSERT INTO mensagens (
            Hora, Sala, Sensor, Leitura, TipoAlerta, Msg,
            HoraEscrita, atuadorAcionado, IDJogo
        )
        VALUES (
            CURRENT_TIMESTAMP, NEW.Sala, 1,
            CONCAT('Odd: ', NEW.numeromarsamisodd, ',
            Even: ', NEW.numeromarsamiseven),
            'movimento', 'ativar gatilho',
            CURRENT_TIMESTAMP, FALSE, NEW.IDJogo
        );
    END IF;
```



END

4. Nome Trigger: **__cria_salas_jogo__**
//Cria as salas na tabela ocupacao labirinto, sempre que é criado um novo jogo.

```
BEGIN
    DECLARE i INT DEFAULT 1;
    WHILE i <= 10 DO
        INSERT INTO ocupacaolabirinto (Sala, IDJogo,
numeromarsamisodd, numeromarsamiseven, tentativas)
            VALUES (i, NEW.IDJogo, 0, 0, 0);
        SET i = i + 1;
    END WHILE;
END
```



Código Stored Procedures implementados

1. Nome SP: **CriarUtilizador**

//Cria um utilizador (jogador) na bd e escreve a informação na tabela utilizador(nome). Além disso dá acesso aos SP's que são possíveis de executar por um jogador. Cria também uma view para o jogador poder visualizar os seus jogos.

```
BEGIN
    -- Criar utilizador MariaDB
    SET @sql = CONCAT('CREATE USER \'', p_nome,
        '\''@\'localhost\' IDENTIFIED BY \'', p_password, '\';');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- Conceder permissão para executar as stored
    procedures
    SET @sql = CONCAT('GRANT EXECUTE ON PROCEDURE
        AlterarDescricaoJogo TO \'', p_nome,
        '\''@\'localhost\';');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    SET @sql = CONCAT('GRANT EXECUTE ON PROCEDURE
        EditarUtilizador TO \'', p_nome, '\''@\'localhost\';');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    SET @sql = CONCAT('GRANT EXECUTE ON PROCEDURE
        CriarJogo TO \'', p_nome, '\''@\'localhost\';');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- Criar a VIEW para os jogos do utilizador
    SET @sql = CONCAT('CREATE OR REPLACE VIEW vw_jogos_',
        p_nome, ' AS
        SELECT IDJogo AS ID, Descricao AS Descricao,
        Estado AS Estado, Pontuacao AS Pontuacao
        FROM jogo
        WHERE JogadorID = (SELECT ID FROM utilizador
        WHERE Nome = \'', p_nome, '\');');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
```



```
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Conceder permissões de SELECT, UPDATE, DELETE na
VIEW de jogos
SET @sql = CONCAT('GRANT SELECT, UPDATE, DELETE ON
vw_jogos_', p_nome, ' TO \'', p_nome,
'\''@\'localhost\';');
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Conceder permissões de INSERT, UPDATE na tabela
jogo
SET @sql = CONCAT('GRANT INSERT, UPDATE ON jogo TO
\'', p_nome, '\''@\'localhost\';');
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Conceder permissões de UPDATE na tabela utilizador
SET @sql = CONCAT('GRANT UPDATE ON utilizador TO \'',
p_nome, '\''@\'localhost\';');
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Criar a VIEW para os próprios dados do utilizador
SET @sql = CONCAT('CREATE OR REPLACE VIEW
vw_utilizador_', p_nome, ' AS
SELECT Nome, Email, Telemovel, Tipo FROM
utilizador WHERE Nome = \'', p_nome, '\';');
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Conceder permissão apenas na VIEW de utilizador
SET @sql = CONCAT('GRANT SELECT ON vw_utilizador_',
p_nome, ' TO \'', p_nome, '\''@\'localhost\';');
PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;

-- Inserir o novo utilizador na tabela da aplicação
INSERT INTO Utilizador (Nome, Telemovel, Email, Tipo)
VALUES (p_nome, NULL, NULL, 'U');

END
```



2. Nome SP: **AlterarDescricaoJogo**

// Altera a descricao de um jogo escolhido.
Apenas se o jogo for do utilizador que está
autenticado no momento, na base de dados.

```
BEGIN
    DECLARE v_idJogador INT;
    DECLARE v_nome VARCHAR(100);

    -- Extrai apenas o nome do user sem o host (ex:
    'grupo20@localhost' → 'grupo20')
    SET v_nome = SUBSTRING_INDEX(USER(), '@', 1);

    -- Busca o ID do utilizador com esse nome
    SELECT ID INTO v_idJogador
    FROM utilizador
    WHERE Nome = v_nome
    LIMIT 1;

    -- Verifica se é o dono do jogo
    IF EXISTS (
        SELECT 1 FROM jogo
        WHERE IDJogo = p_idJogo AND JogadorID =
v_idJogador
    ) THEN
        UPDATE jogo
        SET Descricao = p_novaDescricao
        WHERE IDJogo = p_idJogo;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Permissão negada: utilizador
não é dono deste jogo.';
    END IF;
END
```




3. Nome SP: **CriarJogo**

//Cria dados de um jogo novo na tabela jogo

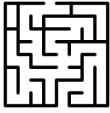
```
BEGIN
    DECLARE v_nome VARCHAR(100);
    DECLARE v_idJogador INT;

    -- Obter o nome do utilizador autenticado na bd
    (sem o host)
    SET v_nome = SUBSTRING_INDEX(USER(), '@', 1);

    -- Obter o ID do utilizador com esse nome
    SELECT ID INTO v_idJogador
    FROM utilizador
    WHERE Nome = v_nome
    LIMIT 1;

    -- Verifica se encontrou o utilizador
    IF v_idJogador IS NOT NULL THEN
        INSERT INTO jogo (Descricao, JogadorID,
Estado)
        VALUES (p_descricao, v_idJogador, 1);

        -- Retorna o ID do jogo recém-criado (isto
seria util para depois executar o script mqtttomysql
através do php)
        SET p_idJogo = LAST_INSERT_ID();
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Utilizador não
encontrado.';
    END IF;
END
```



4. Nome SP: **EditarUtilizador**

//Altera os dados que estão nos argumentos, do utilizador atual.

```
BEGIN
    DECLARE v_nome VARCHAR(100);
    DECLARE v_idUtilizador INT;

    -- Obter o nome do utilizador autenticado na
bd (sem o host)
    SET v_nome = SUBSTRING_INDEX(USER(), '@',
1);

    -- Obter o ID do utilizador com esse nome
    SELECT ID INTO v_idUtilizador
    FROM utilizador
    WHERE Nome = v_nome
    LIMIT 1;

    -- Verificar se encontrou o utilizador
    IF v_idUtilizador IS NOT NULL THEN
        UPDATE utilizador
        SET Telemovel = p_telemovei,
            Email = p_email
        WHERE ID = v_idUtilizador;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Utilizador não
encontrado.';
    END IF;
END
```



5. Nome SP: **CriarJogoAdmin**

//Permite ao admin criar um jogo com o id de um utilizador existente.

```
BEGIN
    -- Verifica se o jogador existe na tabela
    utilizador
    DECLARE jogador_existente INT;

    SELECT COUNT(*) INTO jogador_existente
    FROM utilizador
    WHERE ID = p_JogadorID;

    -- Se o jogador não existir, retorna um erro
    IF jogador_existente = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT
    = 'Jogador não encontrado';
    ELSE
        -- Insere o novo jogo na tabela jogo com
        pontuação inicial 0
        INSERT INTO jogo (Descricao, JogadorID,
        Estado, Pontuacao)
        VALUES (p_Descricao, p_JogadorID,
        p_Estado, 0);
    END IF;
END
```