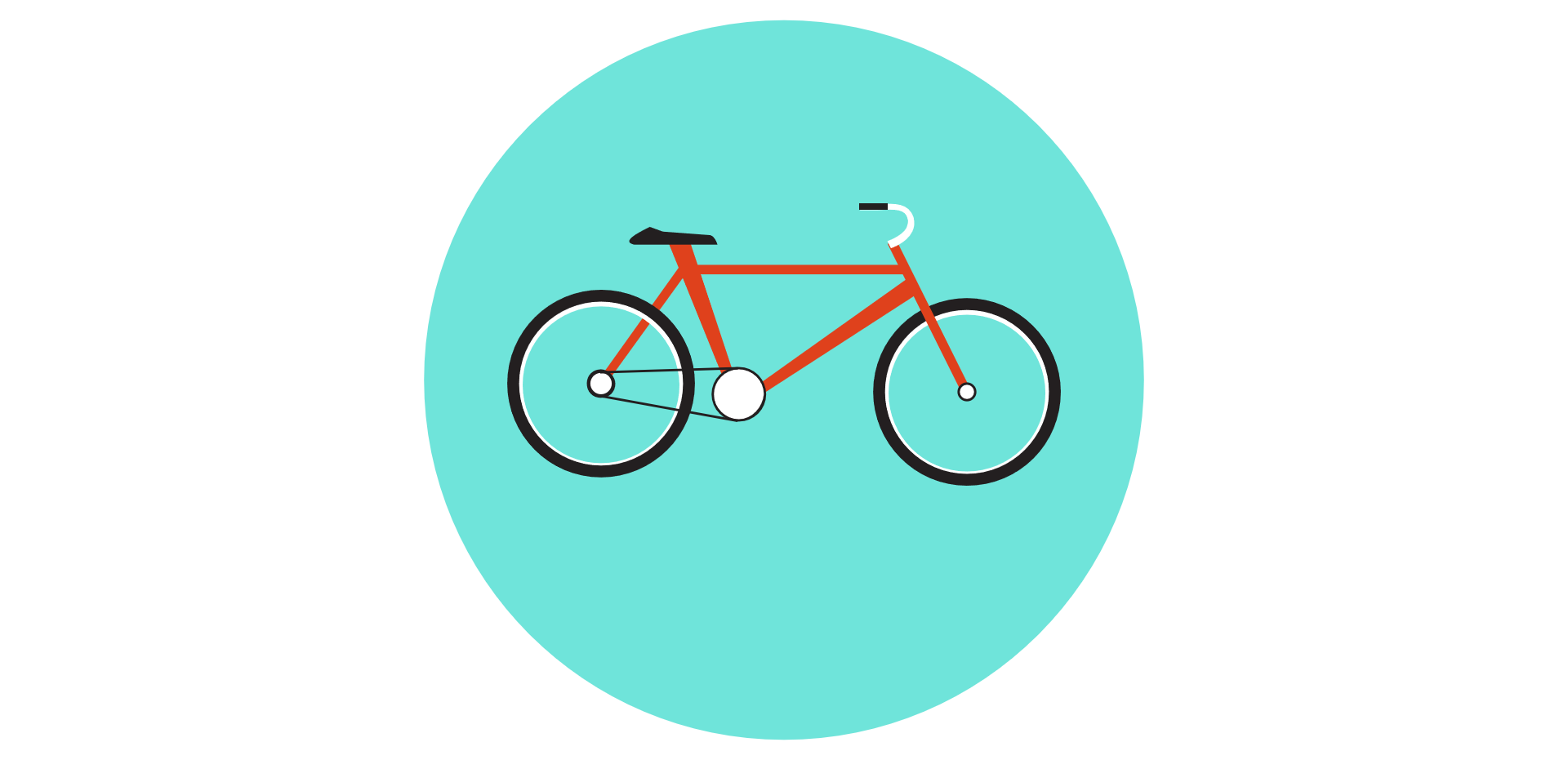
|  |  |
| --- | --- |
| ../../Assuntos%20Gerais/logotipos/estg.png | **Licenciatura em Engenharia Informática**  Escola Superior de Tecnologia e Gestão  *Instituto Politécnico de Viana do Castelo* |
|  |  |



Safe Bike

Documento de especificação e implementação da app

*Computação Móvel – 2018/2019*

01 de Abril de 2019

Constituição da Equipa

|  |  |
| --- | --- |
|  | 12127  João Moreira de Castro  [castrojoao@ipvc.pt](mailto:castrojoao@ipvc.pt) |
|  | 13587  Henrique Maranhão de Sousa  [henriquesousa@ipvc.pt](mailto:henriquesousa@ipvc.pt) |
|  | 14462  Vasco Macedo Freitas  [vasco.freitas@ipvc.pt](mailto:vasco.freitas@ipvc.pt) |
|  | 15178  Pavel Budzko  [pavelbudzko@ipvc.pt](mailto:pavelbudzko@ipvc.pt) |

Docente

|  |  |
| --- | --- |
|  | Sara Maria da Cruz Maia de Oliveira Paiva  [sara.paiva@estg.ipvc.pt](mailto:sara.paiva@estg.ipvc.pt) |

Sumário

[1. Âmbito / Enquadramento do projeto 3](#_Toc187463)

[*2.* Objetivos 3](#_Toc187464)

[*3.* Metodologia de desenvolvimento 3](#_Toc187465)

[4. Arquitetura da solução 3](#_Toc187466)

[5. Análise 3](#_Toc187467)

[5.1 Requisitos 3](#_Toc187468)

[5.2 Mockups 3](#_Toc187469)

[6. Implementação 4](#_Toc187470)

[6.1 Base de dados 4](#_Toc187471)

[6.2 Funcionamento online-offline 4](#_Toc187472)

[6.3 Listagem de Web Services 4](#_Toc187473)

[6.4 Protótipo desenvolvido 4](#_Toc187474)

[7. Conclusões e Melhorias futuras 4](#_Toc187475)

# Âmbito / Enquadramento do projeto

*A aplicação permite que clientes que estejam registados possam alugar bicicletas num determinado ponto de recolha e.g. postos de turismo, stands,etc.*

*Permite ao utilizador definir uma área para a zona que irá utilizar a bicicleta, se ultrapassar essa área o sistema envia um alerta. Durante o aluguer caso haja um movimento abrupto, o cliente recebe um alerta a perguntar se está tudo bem. Se este não responder automaticamente o sistema enviará um sms com a localização exata.*

*Ao fim de 1000 KM o cliente obterá desconto de um aluguer gratuito numa futura reserva, como tambem poderá ver o total das mesmas.*

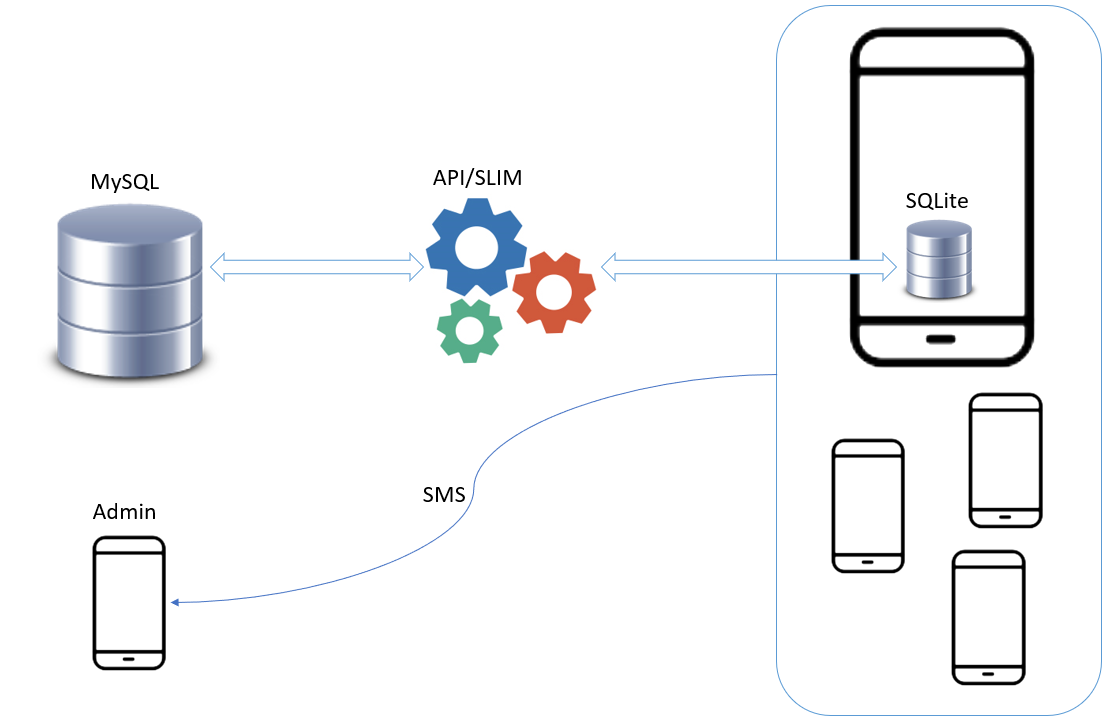
# Objetivos

*Para além do objectivo principal ser um gestor de aluguer de bicicletas, a aplicação deve permitir que o Administrador tenha um controlo sobre as bicicletas que aluga ao nivel de localização/zona e o Cliente uma componente de segurança em caso de queda.*

# Metodologia de desenvolvimento

*<explicar como a equipa se organizou, responsabilidades de cada um, ferramentas de gestão de projeto usadas>*

# Arquitetura da solução



*A nível de base de dados é usada duas tecnologias, tais como, MySql para base de dados central e Sqlite para aplicação móvel. A comunicação entre estas é feita por uma API Rest/Slim que é codificada em Php. A aplicação é codificada em Java. Em termos de alarmistica é usado o SMS.*

# Análise

## Requisitos

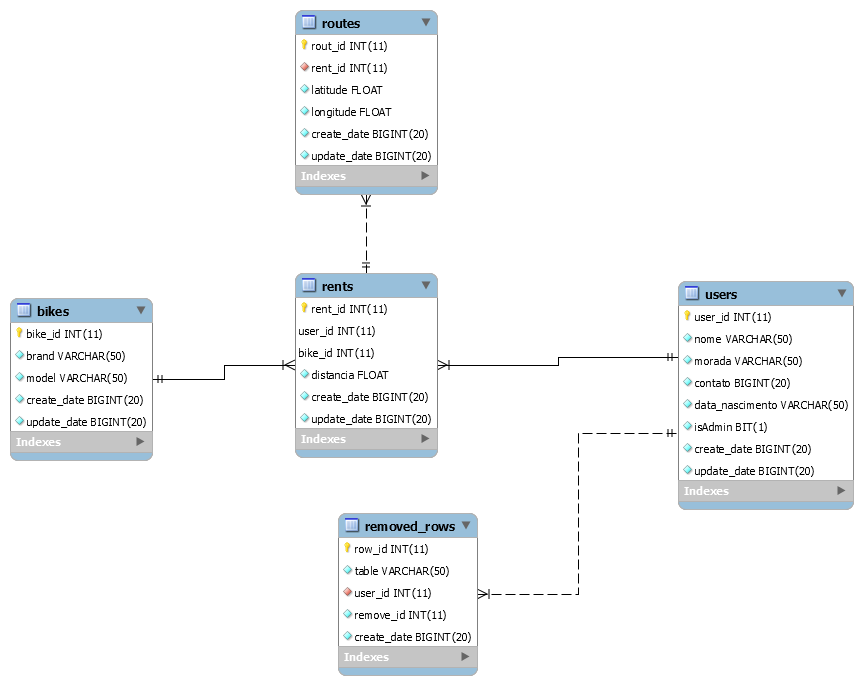
|  |  |  |
| --- | --- | --- |
| *Ator* | *Caso de Uso* | *Descrição* |
| *Cliente* | *Login* | *Se já tiver o registo feito o user pode fazer login na app preenchendo assim os campos “Login” e “Password” e carregar no botão “Entrar”.* |
| *Cliente* | *Criar conta* | *Se não tiver o registo na app o user carrega no botão “Criar Conta”.*  *A seguir procede o registo preenchendo os campos, tais como: “Nome”, “Data nascimento”, “Email”, “Contato” ao terminar o registo carrega no botão “Criar conta”.* |
| *Cliente* | *Reservar bicicleta* | *No ecra principal da app user carrega no icon de “reservar bicicleta” onde pode visualizar uma lista de todas as bicicletas disponíveis para reserva , ou uma mensagem “nenhuma bicicleta disponível”. Caso existem bicicletas o user carrega no botão “reservar” e confirma a reserva carregando em “sim” na alerta que aparece a seguir.* |
| *Cliente* | *Definir a área para a zona que irá utilizar a bicicleta* | *No ecra principal da app user carrega no icon “Área” onde pode definir a área para utilização da bicicleta. área só pode ser definida uma vez e não pode ser alterada. Depois de definir a área carrega em “Definir área” para guardar a definição da área.* |
| *Cliente* | *Alteração abruta da bicicleta* | *Caso a app deteta uma alteração abruta, será enviado um alerta para o user a avisar que houve uma alteração abruta e é necessário confirmar se o user encontra-se bem. Caso o user escolha “Sim” a app prossegue o seu funcionamento normal. Caso escolher “Não” é enviado uma sms ao administrador com a localização da bicicleta. Opção “não” é escolhida automaticamente caso utilizador não escolha nenhuma opção em 2 minutos.* |
| *Cliente* | *Estatísticas/Editar conta* | *No ecra principal da app user carrega no icon de “Conta” onde pode visualizar todas as estatísticas da sua conta, km percorridos, vales de desconto disponíveis e numero de alugueres feitos dessa mesma conta. Também pode visualizar os dados da conta preenchidos no memento do registo. User pode editar os mesmos dados carregando em opção “Editar” e no final confirmar os novos dados preenchidos com o botão “Guardar dados”.* |

## Mockups

|  |  |  |
| --- | --- | --- |
| *Mockup* | *Caso de uso associado* | *Funcionamento.*  *Verificações a efetuar.*  *WS´s a executar.* |
|  | *Login* | - Verificar que o email e password existe na base de dados;  - Se exitir, é efectuado o login;  - Se não tiver conta registada, poderá clicar em criar conta.  - Se não existir conexão à Internet, o utilizador se já tiver conta registada efectuará o login. |
|  | *Criar conta* | *-* Verificar que o campo email, é um email válido;  - Verificar que o campo contacto tem 9 caracteres;  - Verificar que o email não existe na BD;  - Se não existir conexão à Internet, o utilizador cria a conta que posteriormente será sincronizada. |
|  | *Menu* | *-* Verificar se tem sinal GPS e iniciar.  - Se não existir sinal GPS, o utilizador recebe um alerta a indicar o mesmo e não inicia o registo do trajeto. |
|  | *Reservar bicicleta* | *-* Verificar se a bicicleta está disponível.  - Se sim, pode reservar. |
|  | *Definir a área para a zona que irá utilizar a bicicleta* | *-* Verificar se tem sinal GPS .  - Se não existir sinal GPS, o utilizador recebe um alerta a indicar o mesmo e não inicia o registo da zona.  - Gravar a àrea escolhida; |
|  | *Alteração abruta da bicicleta* | *-* Verificar se tem sinal GPS;  *-* Se não existir sinal GPS, vai buscar a última localização gravada.  *- Se ao fim de 2m não responder ao alerta será enviado SMS com a localização.* |
|  | *Editar conta*  *Vale de aluguer grátis*  *Estatísticas* | *-* pode editar os dados pessoais.  - verificar se tem vale grátis  *- ver estatisticas* |

# Implementação

## Base de dados

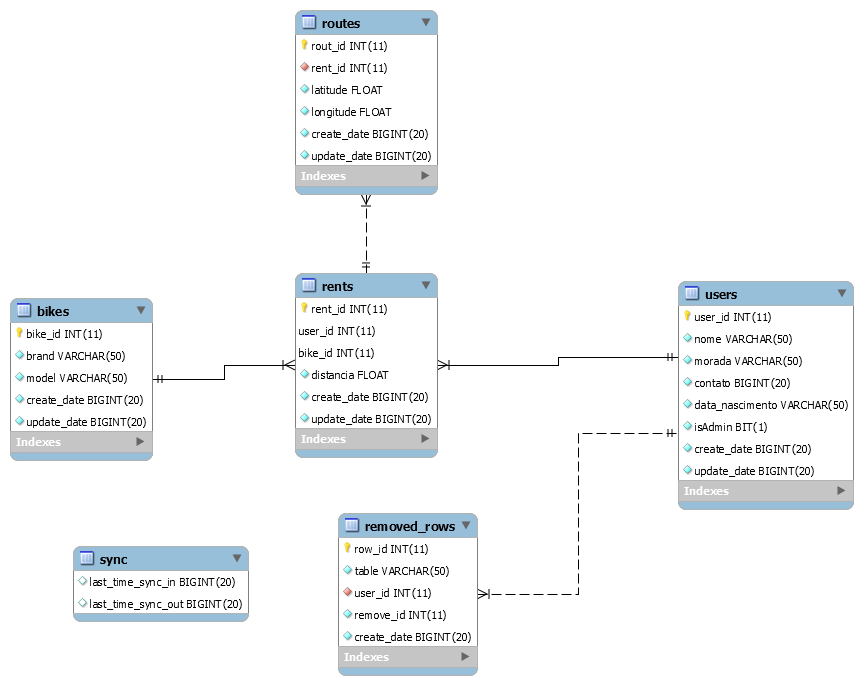
**

*A base de dados remota é contituida por 5 tabelas.*

* *A tabela “users” tem o objectivo de guardar todos os utilizadores bem como o administrador.*
* *A tabela “bikes” é populada pelas bicicletas disponiveis no sistema.*
* *A tabela “rents” guarda os registos de alugueres.*
* *A tabela “routes” regista as áreas conrrespondentes ao aluguer.*
* *A tabela “removed\_rows” ajuda no processo de sincronização.*

## Funcionamento online-offline

*A aplicação funciona em modo offline e tem um serviço de sincronização entre a base de dados local e a remota. O modelo relacinal desta base de dados local é igual ao remoto excpto a tabela de controlo “sync” como se pode ver na figura seguinte:*



*.*

*O seviço de sincronização é composto por duas componentes:*

* ***IN –****este componente tem o objectivo de sincronizar todos os registos que não existem localmente.*
* ***OUT –*** *este componente tem o objectivo de sincronizar todos os registos que não existem remotamente.*

*Para os registos apagados é usada uma tabela de controlo “removed\_rows” que regista todo o que é eliminado.*

*O processo de escolha de registos para o* *CRUD é usado a coluna “update\_date” que está em unix timestamp.*

*É usado uma tabela de apoio “sync” para registar a data quando fez o último sincronismo.*

## Listagem de Web Services

*A base url do web service é a seguinte :* http://{HOST}/safe\_bike/api/v1/

Para o serviço de sincronização temos os seguintes endpoints com os seus inputs:

* **GET**
* /bikes/{date}
* /users/{date}/{user\_id}
* /rents/{date}/{user\_id}
* /routes/{date}/{user\_id}
* /removed/{date}/{user\_id}

Os outputs dos GET serão sempre um JSON da estrutura da tabela.

* **POST**
* /bikes/{date}
* /users/{date}
* /rents/{date}
* /routes/{date}/{user\_id}
* /removed/{date}

Os outputs dos POST serão sempre o ID da tabela e além dos inputs nos parametros é usado o body em JSON como input.

## Protótipo desenvolvido

# Conclusões e Melhorias futuras