Francisco Fernandes (98178), Miguel Reis (108545), Nuno Pinho (108648), Paulo Macedo (102620), Pedro Ferreira (98620)

Versão deste relatório: **2022-05-23**, v1.0

RELATÓRIO – *ELABORATION & CONSTRUCTION*

Construção

Conteúdos

1 Introdução

1.1 Sumário executivo

Este relatório apresenta os resultados da construção do incremento 1, apresentando a arquitetura do sistema e casos de utilização da aplicação.
Os cenários são baseados nos casos de utilização apresentados em apêndice (secção 4)

O primeiro incremento, desenvolvido na Iteração 3, foca a validação da arquitetura proposta. Foram considerados sobretudo as funcionalidades relacionadas com o utilizador.

1.2 Controlo de versões

Quando?	Responsável	Alterações significativas
2023-05-13	Paulo Macedo	Sistema de autenticação e registo da aplicação
2023-05-15	Paulo Macedo	Interface do perfil do utilizador
2023-05-15	Pedro Ferreira	Arquitetura física de instalação
2023-05-16	Pedro Ferreira	Arquitetura do software
2023-05-20	Paulo Macedo	Interface do histórico do utilizador
2023-05-20	Paulo Macedo	Decisões e justificação
2023-05-20	Paulo Macedo	Requisitos com impactos na arquitetura
2023-05-22	Nuno Pinho	Estratégia e estado da implementação
2023-05-23	Nuno Pinho	Especificação dos casos de utilização
2023-05-23	Francisco Fernandes	Objetivos Gerais
2023-05-23	Miguel Reis	Incremento 1

1.3 Referências e recursos suplementares

Para estudar a tecnologia que mais se adequa ao sistema de informação, sendo este um sistema que suporta a geolocalização, foi utilizada a seguinte fonte:

- https://kafka.apache.org/

Também para verificar todas as funcionalidades que a biblioteca **LeafLet** disponibiliza, de forma a verificar a viabilidade da mesma para substituir o Google Maps, foi utilizada a seguinte fonte:

- https://leafletjs.com/

2 Arquitetura do sistema

2.1 Objetivos gerais

Os objetivos gerais para a arquitetura da aplicação de reserva e aluguer de trotinetes são:

1. Acesso fácil e universal: Os utilizadores devem poder aceder à aplicação de reserva e aluguer de trotinetes a partir de qualquer lugar, através de um browser ou de aplicações móveis, sem a necessidade de instalar software específico.
2. Funcionalidades completas: A aplicação deve oferecer um conjunto abrangente de recursos, incluindo reserva de trotinetes e aluguer em grupo ou individual, pagamento, localização e monitorização de trotinetes disponíveis, histórico de aluguer, leitor de QR code.
3. Integração com sistemas externos: A arquitetura deve permitir a integração com sistemas externos, como serviços de pagamento, serviços de mapeamento e localização e serviços de terceiros para verificar a disponibilidade de trotinetes.
4. Desempenho e escalabilidade: A aplicação deve ser projetada para lidar com um grande número de solicitações simultâneas, garantindo um desempenho adequado mesmo em momentos de alta demanda. Além disso, a arquitetura deve ser escalável para suportar o crescimento futuro do número de utilizadores e trotinetes.

5. Confiabilidade e desempenho: A aplicação deve ser confiável e ter um bom desempenho, com tempos de resposta rápidos e mínima ocorrência de erros ou falhas.
6. Segurança: A aplicação deve incorporar medidas de segurança robustas para proteger as informações pessoais dos utilizadores, transações financeiras e garantir a integridade dos dados.
7. Suporte a pagamentos eletrónicos: A aplicação deve integrar-se com plataformas de pagamentos eletrónicos para permitir transações seguras e desmaterializadas. A arquitetura deve ser flexível o suficiente para permitir a substituição do fornecedor de serviços de pagamento sem interrupção das operações.

Estes objetivos gerais orientarão as escolhas de arquitetura para a construção da aplicação de reserva e aluguer de trotinetes, garantindo uma experiência de alta qualidade, integração eficiente e operação robusta.

2.2 Requisitos com impacto na arquitetura

Requisitos	Descrição
Rint.2	O portal da loja deve-se ajustar para ter uma apresentação adequada ao ecrã, designadamente para *smartphones* e *tablets*.
RDes.2	A plataforma deverá fornecer os dados mais recentes sobre a sua frota de trotinetes aos seus utilizadores.
RDes.4	Atualização e manutenção contínua da aplicação para garantir o funcionamento da mesma.
RSeg.2	Toda a informação relativa a um utilizador(histórico de reservas e alugueres de trotinetes, também como o registo do trajeto efetuado) deve ser cifrada.
RExt.1	É necessário visualizar trotinetes e trajetos num mapa

2.3 Decisões e justificação

Tendo em conta os objetivos para a arquitetura, e os requisitos levantados na Análise, foram tomadas as seguintes decisões:

➔ Frontend implementado com o recurso a **HTML** e **CSS**, devido à sua compatibilidade com diversos dispositivos, e à fácil implementação, recorrendo a bibliotecas, tal como a biblioteca ***Bootstrap 5***, que disponibiliza vários componentes para serem implementados, com a possibilidade de alterar o seu aspeto com as classes disponibilizadas. Os membros do grupo também têm bastante conhecimento com esta biblioteca e linguagens, facilitando o desenvolvimento da plataforma.

➔ Para tornar a plataforma interativa e dinâmica, foi utilizado **JS** e **JQuery** (biblioteca em **JS**). O **JavaScript** é bastante versátil facilitando, por exemplo, a interação com botões. Permite criar uma experiência interativa com o utilizador, auxiliando as tarefas de alugar trotinetes e visualização da disponibilidade de trotinetes.

➔ Transmitir informação dos equipamentos para a **API** é necessário uma plataforma rápida, com a capacidade de processar muita informação, sempre com a aptidão de escalabilidade. Portanto, para esse efeito, recorreu-se ao **Kafka**, capaz de enviar e receber conjuntos de dados para um posterior processamento. Para o serviço em questão, isto reflete-se em

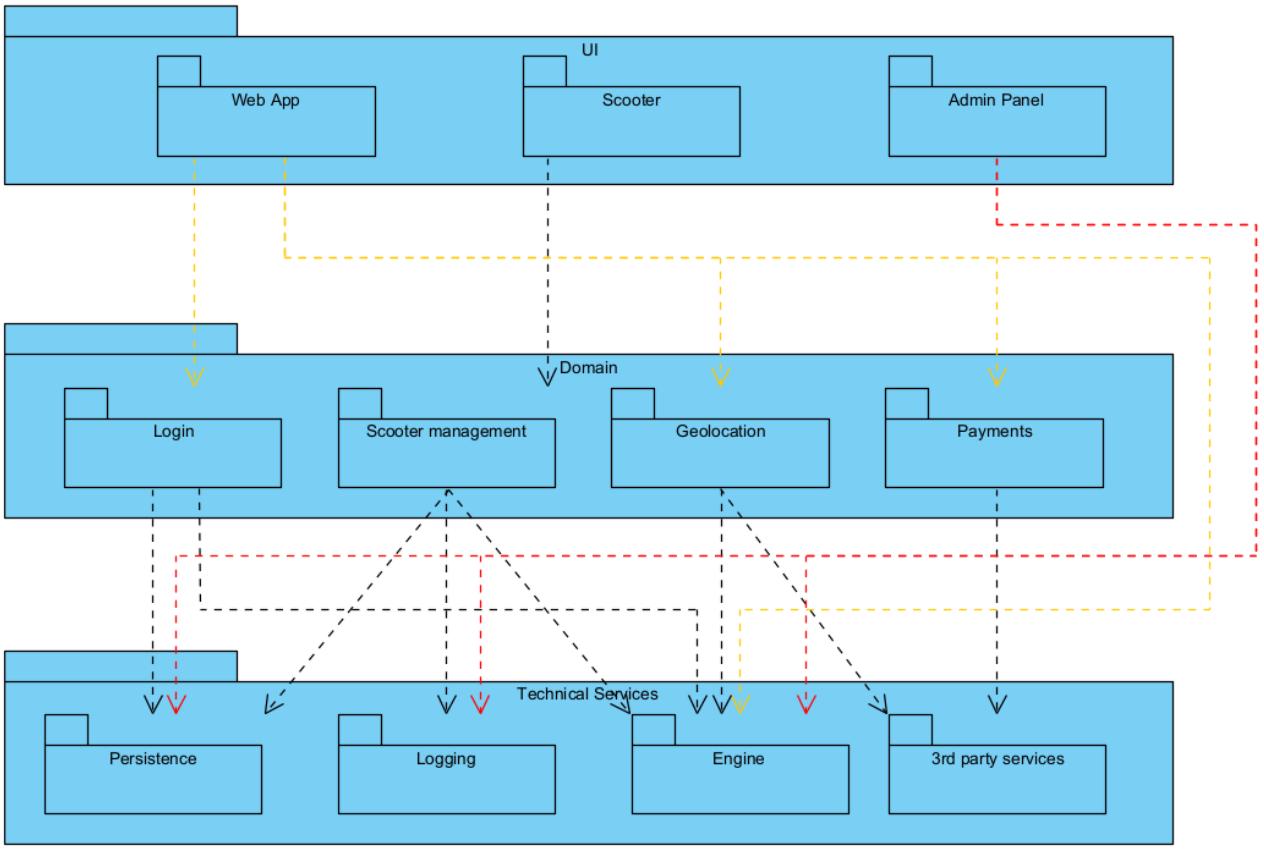
recolher dados sobre a disponibilidade das trotinetes, reservas e alugueres efetuados. Kafka permite lidar com vários eventos de forma eficiente, assegurando a transmissão de dados fulcrais para o bom funcionamento do serviço.

➔ A análise de dados e a sua respetiva monitorização é feita com o recurso à ferramenta, **Prometheus**. Ao integrar o **Prometheus** no serviço de aluguer e reserva de trotinetes, é possível recolher dados relacionados com a performance do sistema(latência, ocorrência de erros, utilização de recursos), que ajuda a identificar possíveis otimizações de performance, e a detetar anomalias. O **Prometheus**, tem a capacidade de gerar alertas, quando certas condições são verificadas. Por exemplo, se o número de aluguéis efetuados sem sucesso exceder um certo limite, esta ferramenta pode ser configurada para alertar este problema em específico, levando à identificação e resolução de problemas antes que possam impactar a todo o serviço.

➔ Para construir uma **REST API**, utilizou-se **Flask**, devido ao facto desta *framework* ser simples e minimalista, fazendo com que seja fácil de usar e compreender. Como os membros do grupo estão familiarizados com **Python**, usar **Flask** para o desenvolvimento de uma **REST API** pode resultar numa melhor eficiência e produtividade no que toca à fase de desenvolvimento.

➔ A biblioteca de mapas digitais será a **LeafLet**. Este serviço possui um plano gratuito e ao contrário de alternativas, como o Google Maps, não mostra watermarks ao longo de todo o mapa, melhorando a experiência. Esta biblioteca fornece também uma grande quantidade de funcionalidades e opções de customização, sendo ideal para mostrar a localização de trotinetes e destinos, facilitando a navegabilidade e visualização do percurso do utilizador.

2.4 Arquitetura do software



A articulação entre os módulos decorre da seguinte forma:

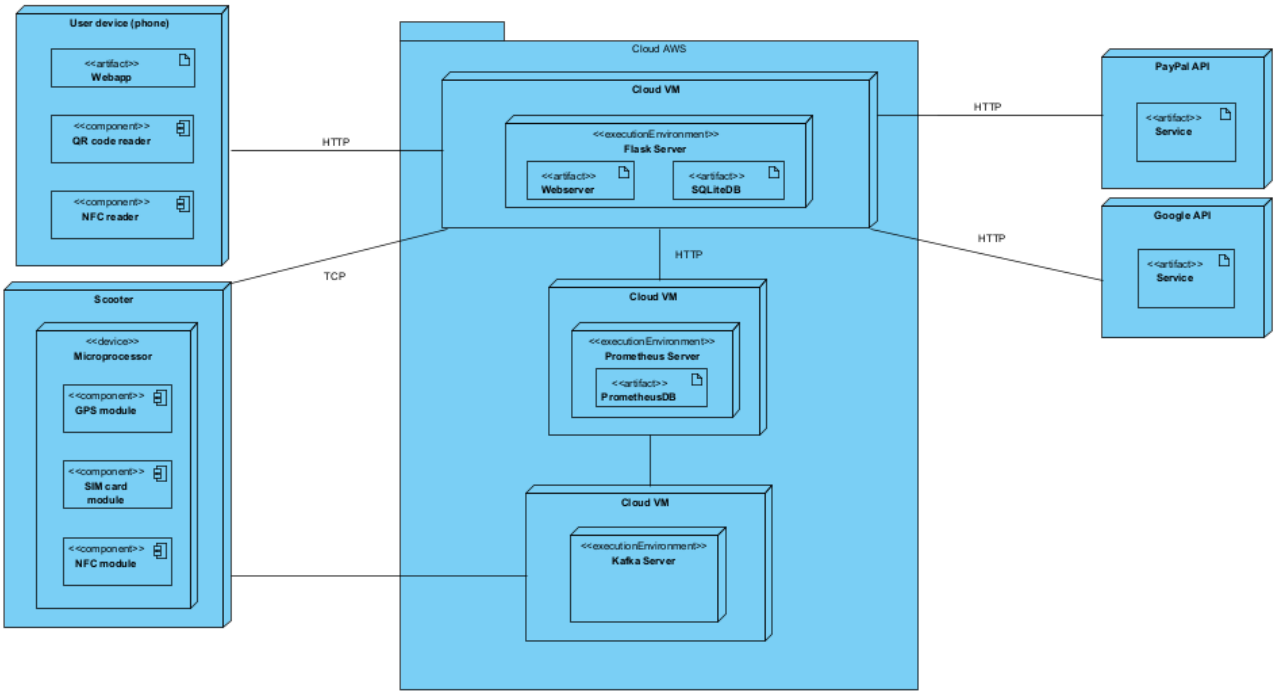
A webapp tem as componentes de login, pagamentos e geolocalização.

A gestão de trotinetes passa por enviar dados para logging, guardando-os numa base de dados.

Para bloquear/desbloquear uma trotinete, é necessário receber um sinal do servidor (engine).

O admin tem acesso a todas as informações guardadas sobre as trotinetes e a webapp, pelo que tem acesso às bases de dados. Também pode bloquear ou desbloquear as trotinetes, por isso também tem acesso aos respetivos endpoints.

2.5 Arquitetura física de instalação



As trotinetes vão estar periodicamente a enviar informações (geolocalização, bateria, estado, etc) para o **Kafka**, que por sua vez as redireciona para o Prometheus (base de dados time-series).

O servidor Flask está a fazer *host* dos *sites* e trata da lógica da aplicação *web*, guardando informações da mesma e dos utilizadores na base de dados (**SQLite**). Quando necessita de informação relativa às trotinetes, envia um pedido **HTTP** para o Prometheus.

Também existe uma ligação websocket entre o servidor **Flask** e as trotinetes para, após ser confirmado o aluguer da trotinete(s), o servidor enviar um sinal para a(s) desbloquear. O mesmo acontecerá no fim da viagem.

Para pagamentos e representação da geolocalização de cada trotinete num mapa, o servidor **Flask** comunica com serviços de terceiros, como a **API** do PayPal e da Google, respetivamente.

3 Incremento 1

3.1 Casos de utilização no Incremento 1

Neste primeiro incremento implementado, o foco esteve na validação da arquitetura proposta, através da implementação da arquitetura do negócio e registo do utilizador. Para isso, selecionámos os casos de uso "Registo do Utilizador" e "Atualização do Perfil". Sendo uma das partes essenciais da aplicação e necessária para o teste e desenvolvimento da mesma, daí a sua implementação em primeiro lugar.

A especificação detalhada dos casos de utilização encontra-se em anexo (secção 4). A partir dessa análise, definiram-se as histórias de utilização a implementar.

3.2 Histórias de utilização selecionadas

As histórias (*user stories*) incluídas nesta interação fazem parte do *backlog* do projeto, acessíveis em https://github.com/PauloMaced0/AS_Project
Histórias incluídas nesta interação:

- A Diana estuda na Universidade de Coimbra e quer arranjar um método de transporte para as suas viagens entre casa e universidade, por isso decidiu registar-se na aplicação "Trotinet".
- Após o login na aplicação, a Diana decidiu que devia mudar a sua palavra-passe para uma mais segura e de forma a preencher os seus dados pessoais, acedeu à sua página de perfil.

História/*use case slice*	**Critérios de aceitação**
A Diana decidiu registar-se na aplicação Trotinete. Após saber da aplicação e da possibilidade de alugar trotinetes para as suas viagens diárias decidiu experimentar.	**Cenário 1: Registo com sucesso** Sendo a primeira vez que usa a aplicação. Estando na página inicial da aplicação. Escolhe a opção de registo e preenche com os seus dados pessoais, escolhendo uma palavra-passe que corresponde aos requisitos. A sua conta é criada com sucesso e pode fazer o Login na aplicação. Surge uma notificação a dizer que a conta foi criada com sucesso e redireciona-a para a página de Login. **Cenário 2: Registo sem sucesso** Estando na página inicial da aplicação. Escolhe a opção de registo e preenche com os seus dados pessoais, escolhendo uma palavra-passe que corresponde aos requisitos. A sua conta não é criada devido a um erro no qual o nome de utilizador ou e-mail já se encontram registados na aplicação.
A Diana decidiu que devia mudar a sua palavra-passe para uma mais segura.	**Cenário 1: Atualização com sucesso**

Após decidir que a aplicação “Trotinet” é bastante útil e prática para a sua vida, decide atualizar os seus dados pessoais e mudar a sua palavra-passe para uma mais segura.	Ao aceder à sua página de perfil, preenche os dados pessoais que restam do primeiro registo na aplicação, primeiro e último nome, a data de nascimento e número de telemóvel. Clica no botão para guardar as alterações e consegue atualizar os seus dados com sucesso. Ainda na mesma página, confirma a sua senha atual e cria uma nova palavra-passe, digita novamente a nova palavra-passe de forma a confirmá-la. Clica no botão para guardar as alterações e consegue atualizar a palavra-passe com sucesso. Surge uma notificação a dizer que a mesma foi atualizada. **Cenário 2: Atualização sem sucesso** Ao aceder à sua página de perfil, preenche os dados pessoais que restam do primeiro registo na aplicação, primeiro e último nome, a data de nascimento e número de telemóvel. Clica no botão para guardar as alterações e consegue atualizar os seus dados com sucesso. Ainda na mesma página, confirma a sua senha atual e cria uma nova palavra-passe, digita novamente a nova palavra-passe de forma a confirmá-la. Clica no botão para guardar as alterações, mas não consegue atualizar a palavra-passe devido à mesma não ter 8 caracteres.

3.3 Estratégia e estado da implementação

Funcionalidades Implementadas:

- Interface para a visualização de alugueres e reservas efetuadas.
- Sistema de log in/sign up com base de dados funcional.
- Perfil do utilizador totalmente funcional.
- Interface para visualizar trotinetes (puramente estática).
- Navegação entre páginas.

Ferramentas Utilizadas:

Frontend:

- **HTML**, **CSS, JavaScript -** formatação e responsividade do website.

Backend:

- **Kafka -** transmissão de dados.
- **Prometheus -** base de dados (time-series).
- **SQLite -** base de dados.
- **Flask -** rest api.
- **LeafLet -** biblioteca de mapas.

Foi utilizado Github para a gestão de tarefas e backlog do projeto.
como também para a gestão do código e controlo de versões.

Apêndice

4 Especificação dos casos de utilização

4.1 Pacote: Conta do Utilizador

4.1.1 Registo do Utilizador

Caso de utilização:	#1: Registo do Utilizador
Versão:	Iteração 1, v2023-05-20
Breve descrição	O Utilizador pretende usufruir do serviço, mas como não possui conta, necessita de primeiro criar uma.
Pré-condições:	O utilizador encontra-se no website do serviço.
Pós-condições	O utilizador pode então utilizar o serviço com a sua conta. Possui também uma página para ver e alterar os detalhes da sua conta.
Fluxo base:	**1. Aceder à página principal do website** O utilizador acede a esta página para usar o serviço. **2. Aceder á página para criar conta** Na caixa de login o utilizador clica em "criar conta" e é dirigido à página. **3. Preencher dados** O utilizador preenche os seus dados escolhendo uma palavra-passe que corresponde aos requisitos. **4. Conta criada** A sua conta é criada com sucesso e pode fazer o login na aplicação.
Fluxos alternativos:	**Passo 2: Já possui conta** Se o utilizador já possuir uma conta não necessita de criar outra, podendo

	assim fazer login direto no serviço.
Exceções:	**Ex1: Sistema ou página do perfil indisponível**
Requisitos especiais:	[Usabilidade] O preenchimento de dados tem que ter em conta os requisitos pedidos.

4.1.2 Alteração de dados da Conta

Caso de utilização:	#2: Alteração de dados da Conta
Versão:	Iteração 1, v2023-05-20
Breve descrição	O Utilizador pretende alterar a sua palavra-passe para uma mais segura, como também pretende preencher os restantes dados para obter uma conta mais completa.
Pré-condições:	O utilizador encontra-se no website do serviço. O utilizador já possui uma conta.
Pós-condições	O utilizador consegue alterar e preencher os dados pretendidos obtendo uma conta mais completa.
Fluxo base:	**1. Entrar na conta** O utilizador entra no website com a sua conta. **2. Aceder à página do perfil** O utilizador clica na aba do perfil dirigindo-se a esta página. **3. Preencher dados** O utilizador preenche os dados escolhendo uma palavra-passe que corresponde aos requisitos. **4. Guardar alterações** O utilizador clica em guardar as alterações feitas. **5. Alterações guardadas** As alterações são guardadas com sucesso.
Fluxos alternativos:	**Passo 5 : As alterações não são guardadas** O utilizador clica em guardar as alterações mas não acontece atualização visto que os requisitos do preenchimento dos dados não foram satisfeitos. **Passo 5.1 : Preencher novamente os dados e guardar alterações** Preencher novamente os dados tendo em conta os requisitos dispostos e guardar as alterações.
Exceções:	**Ex1: Sistema ou página de registo indisponível**
Requisitos	[Usabilidade] O preenchimento de dados tem que ter em conta os requisitos

especiais:	pedidos.