Universidade Aveiro, DETI
IES – Introdução à Engenharia Informática
Universidade Aveiro
[02/01/2023]

Vicente Barros	Nº97787
Diogo Magalhães	Nº102470
Emanuel Marques	Nº102565
Mariana Andrade	Nº103823

RESUMO

O ***Cheku*** é uma aplicação de gestão de veículos que ajuda os utilizadores a gerenciar sua frota de maneira mais eficiente. A aplicação permite ainda acompanhar a localização geográfica dos veículos em tempo real, notifica os utilizadores de pagamentos de seguro e manutenções e permite compartilhar informações com outros membros da família ou da empresa. Além disso, o ***Cheku*** é compatível com uma ampla variedade de veículos e possui uma interface intuitiva e fácil de usar. O ***Cheku*** é ideal para quem precisa gerenciar mais de um veículo e oferece uma gestão colaborativa e eficiente da frota.

Palavras-chaves: Cheku; Agile development; Data stream e Data process

Índice

INTRODUÇÃO

No âmbito da Unidade Curricular de Introdução à Engenharia Informática, do 3º ano do curso de Licenciatura em Engenharia informática, da Universidade de Aveiro, foi solicitada a elaboração de um trabalho para explorar um aplicativo de *multi-layer enterprise-class.*

Para a realização do mesmo, aplicou-se técnicas de Engenharia de Software desde a especificação do produto até à sua implementação, utilizando ferramentas de trabalho colaborativo como o *GitHub* para gerenciar e armazenar o código e o *Jira* para gerenciar tarefas da equipa. Além disso, vamos construir uma arquitetura levando em consideração os requisitos do projeto e discutir a escolha de cada item. Assim, poderemos garantir que a solução atenda às necessidades do projeto de maneira eficiente e escalável.

A metodologia utilizada para realização deste trabalho incidiu-se em pesquisas e apontamentos em sala de aula. As referências inerentes às pesquisas realizadas encontram-se normalizadas de acordo com a Norma APA (American Psychological Association) 7th, com as devidas adaptações inerentes à língua portuguesa.

TEAM

No âmbito da Unidade Curricular de IES, o sistema desenvolvido precisou de uma distribuição do trabalho entre os membros do grupo. De acordo com os requisitos do projeto, foram atribuídas as seguintes funções:

- ***Team Manager:*** Diogo Magalhães
- ***Product Owner:*** Vicente Barros
- ***Devops Master:*** Emanuel Marques
- ***Architect:*** Mariana Andrade

Observação: Apesar das funções atribuídas, todos os membros participaram na criação do projeto.

PRATICAS

BACKLOG

Para gerenciar o desenvolvimento da solução, utilizamos o *software Jira* e seguimos a metodologia *Scrum*, baseada em *Agile*. Criamos vários *Sprints*, geralmente de duas semanas, e distribuímos as tarefas da iteração entre os membros do grupo. Desta forma, conseguimos manter o foco no que precisava ser realizado em cada etapa do projeto e garantir o progresso contínuo da equipa.

WORKFLOW COM *FEATURE-BRANCHING*

De acordo com o que foi solicitado, o processo de desenvolvimento foi estruturado utilizando o *Git* como ferramenta de versionamento de código. Para cada funcionalidade ou correção de *bug* foi criada *branches* cujo nomes descrevem o projeto em questão, o tipo de trabalho realizado ou o nome da *feature*.

Sempre que uma *feature* estava concluída, era criada uma *Pull Request* para que outros membros da equipa pudessem revisar o código e verificar se estava de acordo com as boas práticas de desenvolvimento.

Além disso, para gerenciar o desenvolvimento e acompanhar o progresso, utilizamos a ferramenta *Jira* interligada com o *Git*, atribuindo *commits* específicos a tarefas. Dessa forma, conseguimos garantir a qualidade do código e a eficiência na execução das tarefas.

DEPLOYMENT

O sistema foi implementado com o objetivo de separar as responsabilidades em diferentes containers Docker. Par agilizar a execução do sistema na máquina virtual (VM), utilizamos um *Docker Compose*.

O sistema pode ser acedido pelo seguinte link: http://192.168.160.225/.

CONCEITO DO PRODUTO

VISÃO

Os carros são um meio de transporte, lazer e expressão fundamental na vida de muitas pessoas. No entanto, gerenciar um carro pode ser complexo, pois envolve a gestão de despesas, manutenções e localização.

O *Cheku* é uma aplicação de gestão de veículos que foi desenvolvida para ajudar os usuários a gerenciar sua frota de veículos de maneira mais eficiente e prática. Com a *Cheku*, é possível acompanhar a localização geográfica dos veículos em tempo real, agendar alertas para pagamentos de seguro e manutenções e compartilhar informações com outros membros da família ou da empresa.

A aplicação possui uma interface intuitiva e fácil de usar, e é compatível com uma ampla variedade de veículos. Além disso, ela oferece uma série de recursos adicionais, como relatórios e gráficos, que podem ser úteis para a gestão da frota.

O *Cheku* é ideal para pessoas que precisam gerenciar mais de um veículo, seja para uso pessoal ou empresarial. Além de economizar tempo e dinheiro, o uso da aplicação permite uma gestão mais eficiente e colaborativa da frota, o que pode ser uma vantagem significativa em várias situações.

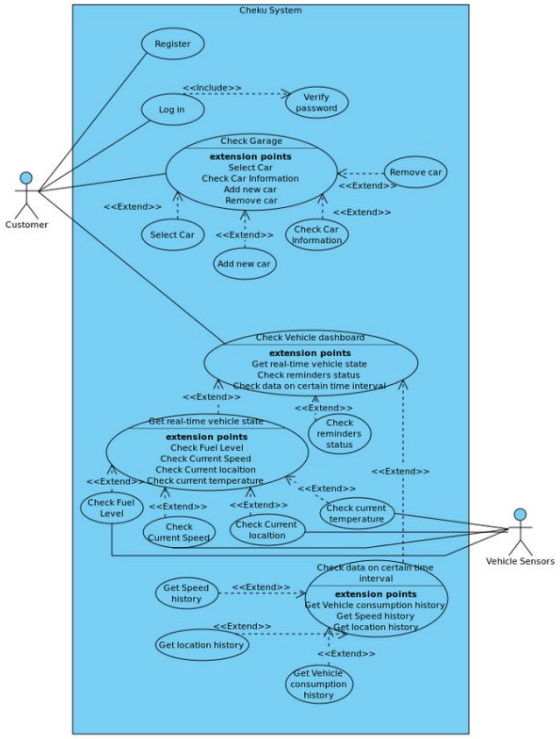

Figura 1 Diagrama de casos de uso

Por exemplo, imagine que você é dono de uma frota de veículos de entrega e precisa manter o controle das manutenções e dos pagamentos de seguro. Com o *Cheku*, é possível agendar alertas para esses compromissos e acompanhar a localização dos veículos em tempo real, o que pode ajudar a prevenir problemas futuros e garantir que a frota esteja sempre em ótimo funcionamento.

Em resumo, o *Cheku* é uma solução completa e inovadora para gerenciamento de veículos, que oferece uma série de benefícios para os usuários.

PERSONAS

Uma *Persona* é uma personagem criada para representar o público-alvo de um sistema ou produto. No nosso sistema, criamos três personas para ilustrar as possíveis utilizações: **Dário Miranda,** um indivíduo apaixonado por carros e que deseja um sistema centralizado para gerenciar sua frota; **Alexandra Silva**, uma mulher que deseja controlar os consumos da frota de veículos da sua família; e **ECNELID LDA,** uma *start-up* de tecnologia que deseja garantir o uso adequado dos veículos da empresa pelos seus funcionários.

Dário Miranda, 28 anos, mora em Ermesinde e trabalha como corretor na empresa *Euronext*. Ele é um apaixonado por carros e possui uma coleção de carros esportivos em várias cidades do país, como Portimão, Lisboa e Bragança. Ele deseja um sistema centralizado para gerenciar sua frota.

Alexandra Silva, de 45 anos, mora com a sua família em Ílhavo e tem uma filha mais velha, *Luísa*, de 21 anos, que precisa usar um dos carros da família para ir à universidade todos os dias. Devido à instabilidade dos preços dos combustíveis fósseis, Alexandra deseja controlar os consumos de combustível da frota de veículos da sua família e conseguir controlar por onde sua filha viaja.

Empresa cujo nome estamos legalmente impedidos de dizer (ECNELID LDA.), é uma *start-up* de soluções tecnológicas com sede em Lisboa, criada em 2018. A empresa possui cerca de 20 colaboradores e 5 carros para facilitar o deslocamento para reuniões. A administração da **ECNELID LDA.** Deseja garantir o uso adequado dos veículos da empresa pelos seus funcionários, assegurando que eles não estão consumindo excesso de combustível ou usando os veículos para tarefas pessoais. Além disso, a empresa quer controlar o uso dos carros para garantir que estão sendo utilizados de maneira correta.

CENÁRIOS PRINCIPAIS

Cenário 1: O *Dário* precisa gerenciar a sua coleção de carros pessoais.

O *Dário* é um jovem que possui uma pequena coleção de carros e precisa gerenciá-los de maneira eficiente. Ele quer ter uma visão geral das informações de cada carro, incluindo marca, modelo, data de seguro ou inspeção, entre outros dados e garantir que todos os automóveis estejam com a manutenção e seguro em dia. Para isso, o Dário utiliza a aplicação de gestão de veículos *Cheku*.

Ele inicia a sessão na aplicação e seleciona na aba dos veículos visualizando a informação do carro selecionado, podendo editar e remover os seus veículos. Além disso, o sistema oferece a funcionalidade de alerta de pagamentos e manutenções.

Desta forma, e com recurso as outras funcionalidades oferecidas pela aplicação o *Dário* consegue gerenciar os seus carros de maneira eficiente e se manter atualizado sobre o status do veículo.

Cenário 2: Monitorização do carro em tempo real.

A *Luísa*, filha da *Alexandra*, decidiu viajar e utilizou um dos carros dos seus pais. Como a *Alexandra* quer ter acompanhamento da localização do veículo durante a viagem, ela utiliza a aplicação de forma a monitorizar o carro em tempo real.

Estando com sessão iniciada na aplicação e tendo escolhido o carro com que a sua filha viaja, ela consegue ter acesso aos dados do carro em tempo real, selecionando a aba de *live*. Desta forma, a *Alexandra* acompanha o percurso da *Luísa* e tem a certeza de que está segura, dando lhe uma maior tranquilidade e segurança enquanto a *Luísa* viaja, pois tem acesso à localização do veículo em tempo real e se necessário pode tomar medidas adequadas em caso de acidente ou avaria.

Cenário 3: Gestão de viagens da empresa

O gerente da *ECNELID LDA* pretende ter uma visão detalhada do uso de seus veículos, incluindo informações sobre as últimas viagens realizadas, os consumos de combustível associados e outros dados.

Para atender a essa necessidade, o gerente utiliza a aplicação de gestão de veículos *Cheku*. Ele pode aceder ao histórico de viagens realizadas na última semana ou último mês, obtendo os diferentes trajetos efetuados e um gráfico de consumo de combustível.

Para tal, ele inicia a sessão, seleciona o carro que pretende ter a informação e o filtro que lhe é mais relevante visualizar.

Assim, o gerente pode monitorar a localização dos veículos da empresa, podendo ser útil em caso de perda ou roubo, oferecendo-lhe mais tranquilidade e segurança ao gerenciar os carros da empresa.

Cenário 4: Notificações, alertas e Lembretes

O Dário é um utilizador da aplicação *Cheku*, que possui alguns carros para uso pessoal e de coleção e procura ter uma ajuda na marcação de manutenções, de pagamentos de seguros e no caso de avarias.

Para atender a essa necessidade, o Dário utiliza a aplicação de forma a ter acesso a notificações pré-enviadas pelo sistema em forma de lembretes ou alertas de falhas, como por exemplo, avaria de uma luz, limite de pagamento do seguro. Isto lhe permite manter o seu veículo sempre em ótimo funcionamento e evitar problemas futuros.

Cenário 5: Compartilhamento de informações e acesso de membros da família ou da empresa

A *Alexandra* é um utilizador que gere os carros da sua família e deseja compartilhar informações e dados eficientes com restantes membros da família. Ela quer que um dos seus filhos tenha acesso a informação do grupo.

Para tal, a *Alexandra* pode compartilhar com os restantes membros da família um *code* que permite que o seu filho quando se registar na aplicação possa entrar diretamente no grupo e aceder as informações já existentes. Este *code* pode ser visualizado apenas na conta do administrador do grupo juntamente com as informações do grupo e do carro.

Desta forma, a aplicação permite-lhe ter mais apoio na gestão dos carros e na flexibilidade para dividir tarefas e responsabilidades.

NOTAS DE ARQUITETURA

PRINCIPAIS REQUESITOS E RESTRIÇÕES

O **Cheku** é uma aplicação *web*, pelo que tem um conjunto de requisitos e restrições particulares, tais como:

- ✓ Todos os utilizadores devem ser corretamente identificados pelo login, utilizando um *username* e uma *passsword*, de forma a aceder aos seus dados, não podendo aceder dados de outros. Caso contrário será lhe permitido criar uma conta para si;

- ✓ A *aplicação* deve funcionar num browser, de forma a ser acedida através da maioria dos dispositivos eletrónicos, que possuam ter uma conexão à internet;

- ✓ Os utilizadores devem conseguir poder aceder a todas as operações CRUD dos veículos, de forma a permitir um gerenciamento dos seus veículos no sistema;

- ✓ O sistema deve ser capaz de gerar e consumir dados automaticamente, simulando o que aconteceria num veículo em movimento. (velocidade, temperatura do motor, nível/quantidade de combustível, localização, etc.);

- ✓ O *cheku* deve ser capaz não apenas de ver dados gerados em tempo real, mas também de armazená-los, para que os usuários possam aceder a esses dados mais tarde, por exemplo, para gerenciamento de gastos de uma viagem, ver distância percorrida, a quantidade de combustível gasto a média de RPM do carro, etc.;

VISTA DA ARQUITETURA

Cheku: O núcleo do sistema será construído através de um serviço baseado em *Spring Boot.* Este serviço é composto por uma camada *REST Controller*, que permitirá consultar diversos dados do sistema para serem posteriormente apresentados na aplicação *web*, a camada do modelo de dados, que vai permitir o mapeamento dos dados na base de dados para objetos *java* a serem tratados e processados na camada de lógica de negócio.

Aplicação Web: Escolheu-se a framework de *JavaScript React* para construir a aplicação web do sistema devido às vantagens de desenvolvimento que esta face ao desenvolvimento de aplicações dinâmicas.

Geração de dados: Idealmente haveria sensores que iriam recolher dados. No entanto, foi definido que os dados consumidos pelo sistema seriam gerados através do *Python*, simulando os sensores e as informações que recolhem.

Aquisição de dados: Foi definido *RabbitMQ* como *message broker*, com o objetivo de enviar dados gerados de modo a serem guardados e processados pelo sistema e de atualizar dados que ao longo do tempo poderão sofrer alterações por parte dos utilizadores.

Persistência dos dados: Para termos persistência dos dados, precisamos de ter uma base de dados, por isso optamos por escolher *MySQL*, por ser uma boa alternativa no mercado e ajustar-se ao nosso modelo de dados.

Interação com o utilizador: primeiramente uma *aplicação web*. No entanto, mais tarde poderá haver eventualmente uma *aplicação móvel.*

Ainda é importante referir que cada um dos componentes referidos irá correr em separado dentro de um *docker container*. Pelo que desta forma estaremos a isolar os componentes e permite ter todo este ambiente em máquinas fisicamente separadas tornando num sistema distribuído.

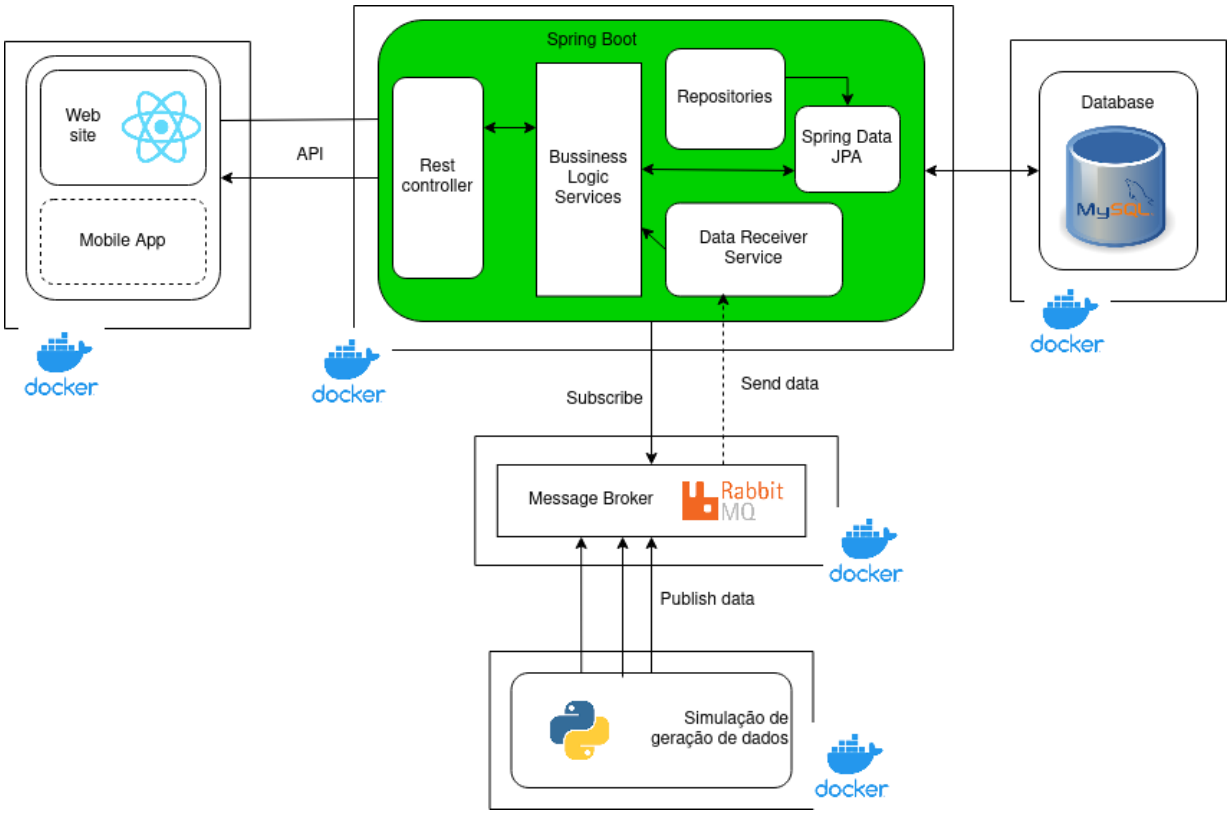

Figura 2 Arquitetura do sistema

MODULE INTERACTIONS

A nossa arquitetura consiste na interação de vários módulos com um núcleo principal em *spring-boot*. Esses módulos podem ser totalmente independentes entre si, ou ter alguma conexão, o gerador de dados relaciona-se com o *message broker*, mas nunca interfere com a interface web, por exemplo.

O gerador de dados é criado por um script *Python* que simulará vários sensores que captam informações e as envias para um tópico do *RabitMQ*. Este processo é contínuo e existe um comando que permite subscrever um tópico desses tópicos. Os dados são depois enviados para serviços de processamento de informação que os filtram, precessão e limpam antes de invocar métodos de acesso à base de dados de forma a armazená-los de forma persistente.

O utilizador controla a maioria das ações através da interface *web*, onde pode realizar operações como adicionar, alterar, remover um veículo, ou visualizar informações. A interface web se comunica com a lógica de negócios através da API do sistema para recolher e apresentar dados ao utilizador. Se a operação for de alteração,

remoção ou adição de dados, a resposta da aplicação pode não ser complexa. No entanto, muitas vezes a lógica de negócio precisará de aceder a base de dados e, nesse caso, se comunicar com o *controller* responsável pela função, que envia a resposta para o *controller* principal depois de receber o resultado das *queries*. O *controller* principal, por sua vez, envia a resposta à interface *web* como resposta a um *endpoint* da API.

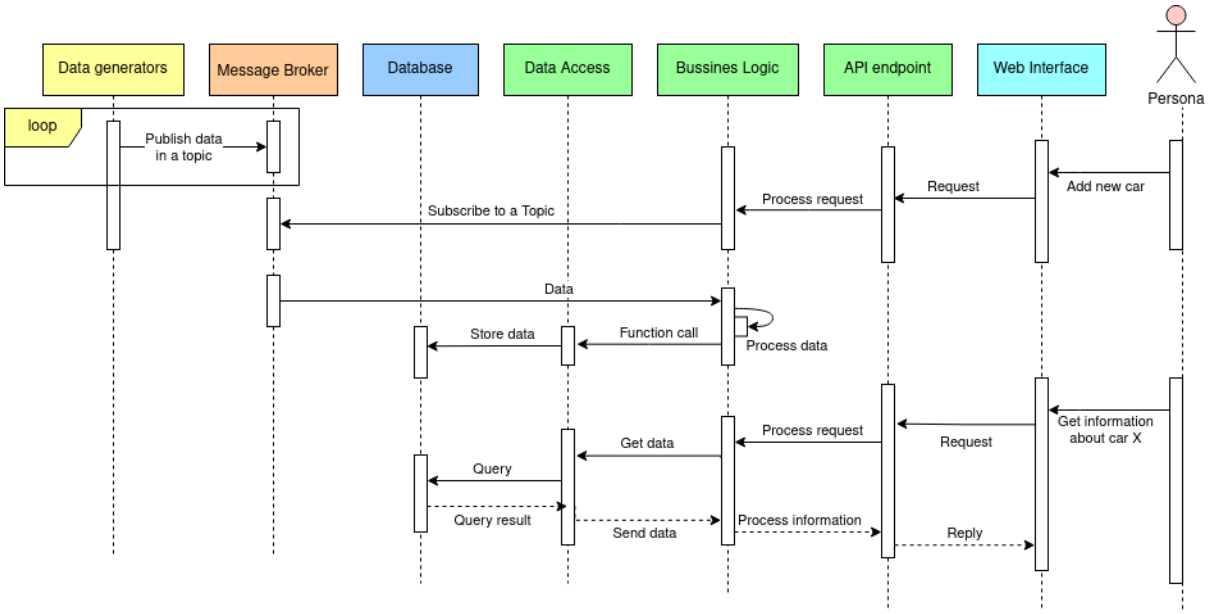

Figura 3 Diagrama de sequência

PERSPETIVA DA INFORMAÇÃO

Para atender às necessidades do projeto e garantir a sua complexidade, decidimos utilizar uma base de dados relacional para gerenciar o elevado número de relacionamentos entre as entidades. Consideramos o MySQL como a opção mais adequada, mas outras alternativas seriam o PostgreSQL e o MariaDB.

Além disso, uma base de dados relacional oferece vantagens como a facilidade de estabelecer relacionamentos entre as entidades, a possibilidade de usar linguagens de consulta para aceder e manipular os dados de maneira eficiente e a garantia da integridade dos dados, evitando inconsistências e garantindo a consistência das informações armazenadas.

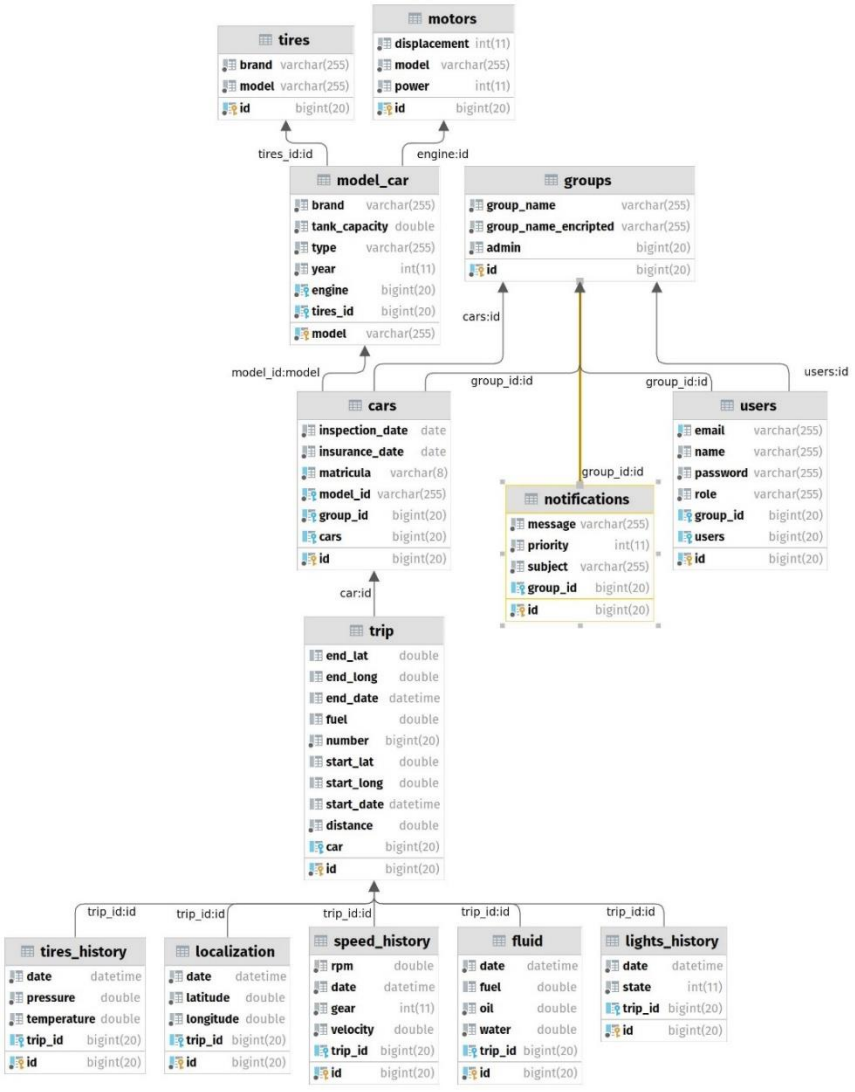

Figura 4 - Diagrama UML

Para a base de dados relacional, temos as seguintes entidades:

***Users*:**

Entidade de utilizador responsável por interagir com o aplicativo *cheku*. Ela contém informações básicas sobre os usuários, *como nome, email e passsword*. Os campos de email e nome são campos únicos.

***Groups*:**

Entidade responsável por integrar diferentes utilizadores em grupos. Sendo útil para permitir que os usuários compartilhem informações e dados eficientes entre si, garantindo uma gestão eficiente e colaborativa de um conjunto de carros.

Cars:

Entidade responsável por armazenar informações sobre os veículos, como *matrícula*, modelo de carro(*model_car*), *data de seguro* e outros detalhes. Os usuários podem adicionar, editar e remover veículos através da aplicação.

Model_car:

Entidade que armazena informações específicas sobre o modelo de carro, como *ano de fabricação, motor, pneus, marca* e outros detalhes.

Tires:

Entidade que armazena informações sobre os pneus, incluindo *marca, modelo.*

Motors:

Entidade que armazena informações sobre os motores dos automóveis, como *potência, cilindrada* e *modelo.*

Trip:

Entidade responsável por armazenar informações sobre as viagens dos automóveis, como velocidade média, distância percorrida e quantidade de combustível gasta.

***Fluid*:**

Entidade que armazena informações sobre os fluidos de um carro, como óleo, água e combustível.

Localization:

Entidade que armazena informações sobre a localização geográfica de um carro, incluindo a latitude e longitude.

Notifications

Entidade que armazena informações sobre as notificações enviadas pelo aplicativo *Cheku*, incluindo o conteúdo da notificação, o destinatário e a data de envio.

As entidades ***Tires_history, Speed_history Ligths_history,*** armazenam informações sobre o histórico dos pneus, da velocidade e das luzes, respetivamente, para além o atributo que a identifica, está associado a um carro e uma data de quando a que a informação foi recolhida. Estas informações podem ser usadas para monitorar o desempenho do veículo e detetar possíveis problemas ou falhas.

CONCLUSÃO

A realização deste projeto, no âmbito de Introdução Engenharia de Software permitiu a aprendizagem da importância do trabalho em equipa em projetos.

O uso de diversas ferramentas de gestão de projetos, tornou possível a aprendizagem da utilização de cada uma das mesmas e a sua importância no desenvolvimento de um projeto. A utilização do GitHub permitiu a aprendizagem, não apenas como uma ferramenta de armazenamento e partilha de código, mas também como uma ferramenta que nos ajuda a controlar as versões do nosso código, as tarefas que cada membro da equipa realizou e a controlar e impedir a entrada de código mal construído e/ou malicioso através do pull requests e revisões de código. A utilização do Jira permitiu a aprendizagem que nos ajuda a gerir as tarefas que cada membro da equipa deve realizar, a sua prioridade e o seu estado de realização. Com o Jira ligado ao GitHub, foi possível ter uma rápida ligação entre as tarefas que cada membro da equipa realizou e o código que foi escrito para a realização dessas tarefas através dos códigos da tarefa.

Respetivamente ao desenvolvimento do projeto em si, utilizamos diversas ferramentas como o React, o MySQL, o RabbitMQ e o Spring Boot, o que nos ajudou a descobrir algumas destas ferramentas e/ou a melhorar nas outras, e ainda utilizamos o Docker para a criação de containers, o que foi um desafio para nós, pois tivemos de aprender a construir aplicações que comunicam entre si através de containers que poderiam se encontrar em diferentes redes e máquinas.

No caso do deste nosso projeto, foi utilizada a metodologia Agile. Esta metodologia revelou-se uma metodologia que pode reduzir o tempo de produção de um produto, uma maior qualidade de produto, uma redução de risco, uma maior flexibilidade das prioridades, uma maior transparência para com o cliente, uma melhor otimização das tarefas e uma maior satisfação do cliente.

Apesar de todas as dificuldades no desenvolvimento deste projeto, o mesmo revelou-se um estímulo positivo para o grupo.

REFERÊNCIAS E RECURSOS

API DOCUMENTATION

POSTMAN, M Andrade, V Barros, D Magalhães, E. Marques, (2023). Documentação API. https://documenter.getpostman.com/view/13973483/2s8YzMY5S1

REFERENCES

Agustina Aguilera. (n.d.). Dockerize a Vite/ReactJS application. Disponível em: https://dev.to/agustinaguilera/dockerize-vitereactjs-application-6e1.

Baeldung. (n.d.). Dockerizing a Spring Boot Application. Disponível em: https://www.baeldung.com/dockerizing-spring-boot-application.

Baeldung. (n.d.). Hibernate One-To-Many Relationship Mapping Example. Disponível em: https://www.baeldung.com/hibernate-one-to-many

Baeldung. (n.d.). Persisting Enums in JPA. Disponível em: https://www.baeldung.com/jpa-persisting-enums-in-jpa

DaisyUI. (n.d.). Documentation. Disponível em: https://daisyui.com/ .

Dockerize. (n.d.). Docker Spring Boot Guide. Disponível em: https://dockerize.io/guides/docker-spring-boot-guide.

Java Guides (2022) Spring Boot RabbitMQ Multiple Queues Disponível em: https://www.javaguides.net/2022/07/spring-boot-rabbitmq-multiple-queues.html.

Java to Dev. (n.d.). Spring Boot JWT Authentication. Disponível em: https://javatodev.com/spring-boot-jwt-authentication/.

Mapbox. (n.d.). Documentation. Disponível em: https://www.mapbox.com/.

RabbitMQ. (n.d.). Tutorial one: “Hello World!” Disponível em: https://www.rabbitmq.com/tutorials/tutorial-one-python.html.

React. (n.d.). Documentation. Disponível em: https://reactjs.org/.

Recharts. (n.d.). Documentation. Disponível em: https://recharts.org/en-US/.

Tailwind CSS. (n.d.). Documentation. Disponível em: https://tailwindcss.com/.

Ten Mile Square. (n.d.). Spring Boot JPA Relationship Quick Guide. Disponível em: https://tenmilesquare.com/spring-boot-jpa-relationship-quick-guide/