s

Projeto Temático em Desenvolvimento de Aplicações

Relatório Final

**Gestão de Tráfego - Radares**

|  |  |
| --- | --- |
|  |  |
| Grupo 5: | Austin Foote nº84935  Bruno Ferreira nº95291  Carolina Tavares nº92658  Diogo Oliveira nº95182  Maria Nobre nº92659 |

1ªSemestre Ano Letivo 2019/2020

Projeto Temático em Desenvolvimento de Aplicações

Relatório Final

**Gestão de Tráfego - Radares**

|  |  |
| --- | --- |
|  |  |
| Grupo 5: | Austin Foote nº84935  Bruno Ferreira nº95291  Carolina Tavares nº92658  Diogo Oliveira nº95182  Maria Nobre nº92659 |

Orientador: Joaquim Ferreira

1ªSemestre Ano Letivo 2019/2020

Resumo

Índice

[Índice de tabelas IV](#_Toc29515730)

[1 Introdução 1](#_Toc29515731)

[1.1 Visão geral do sistema 2](#_Toc29515732)

[1.2 Cliente 3](#_Toc29515733)

[1.3 Objetivos 3](#_Toc29515734)

[2 Planeamento 4](#_Toc29515735)

[3 Modelo de requisitos 5](#_Toc29515736)

[3.1 Requisitos funcionais 5](#_Toc29515737)

[3.2 Restrições e requisitos não funcionais 6](#_Toc29515738)

[3.2.1 Requisitos de interface e facilidade de uso 6](#_Toc29515739)

[3.2.2 Requisitos de desempenho 7](#_Toc29515740)

[3.2.3 Requisitos de segurança e integridade dos dados 7](#_Toc29515741)

[4 Modelo de Casos de Utilização 8](#_Toc29515742)

[4.1 Visão geral 8](#_Toc29515743)

[4.2 Atores 9](#_Toc29515744)

[4.3 Descrição dos casos de utilização 9](#_Toc29515745)

[4.3.1 [Estatísticas de Tráfego #1] 10](#_Toc29515746)

[4.3.2 [Configuração Sistema #2] 11](#_Toc29515747)

[4.3.3 [Consultar informações do tráfego #3] 12](#_Toc29515748)

[4.3.4 [Autenticação #4] 12](#_Toc29515749)

[4.4 Cobertura de requisitos 13](#_Toc29515750)

[5 Diagrama de Classes 14](#_Toc29515751)

[6 Diagrama de Entidades 15](#_Toc29515752)

[7 Implementação da Base de Dados 16](#_Toc29515753)

[7.1 Criação das tabelas 16](#_Toc29515754)

[7.2 Criação das views 16](#_Toc29515755)

[7.3 Filtração e inserção dos dados do radar na base de dados 16](#_Toc29515756)

[8 Principais Métodos a Implementar 17](#_Toc29515757)

[9 Testes Unitários 19](#_Toc29515758)

[10 Implementação da aplicação no netbeans 20](#_Toc29515759)

[10.1 Aplicação Cliente – Servidor 20](#_Toc29515760)

[10.2 Conexão à base de dados – JDBC 20](#_Toc29515761)

[10.3 Conexão à base de dados – JDBC 20](#_Toc29515762)

[10.4 Criação interfaces em JFrame 20](#_Toc29515763)

[10.5 Implementação classes e métodos 20](#_Toc29515764)

[10.5.1 Mostrar estatísticas simples 20](#_Toc29515765)

[10.5.2 Mostrar histórico 20](#_Toc29515766)

[10.5.3 Autenticação 20](#_Toc29515767)

[10.5.4 Registo de Entidades 20](#_Toc29515768)

[11 Análise dos Resultados 23](#_Toc29515769)

[12 Reflexão Crítica e Conclusão 24](#_Toc29515770)

[13 Fontes e material de referência 26](#_Toc29515771)

[Anexos 27](#_Toc29515772)

[Anexo A – Planeamento: 28](#_Toc29515773)

**Índice de figuras**

[Figura 1 - Diagrama de Casos de Utilização 8](file:///C:\Users\Maria%20Nobre\Documents\Git\Maquinas\Relatório\RelatórioPTDA.docx#_Toc29515774)

[Figura 2 - Diagrama de Atividades – Estatísticas de Tráfego 10](file:///C:\Users\Maria%20Nobre\Documents\Git\Maquinas\Relatório\RelatórioPTDA.docx#_Toc29515775)

[Figura 3 - Diagrama de Atividades – Configuração Sistema 11](file:///C:\Users\Maria%20Nobre\Documents\Git\Maquinas\Relatório\RelatórioPTDA.docx#_Toc29515776)

[Figura 4 - Diagrama de Classes 14](file:///C:\Users\Maria%20Nobre\Documents\Git\Maquinas\Relatório\RelatórioPTDA.docx#_Toc29515777)

# Índice de tabelas

[Tabela 1 - Requisitos Funcionais 5](#_Toc29515778)

[Tabela 2 - Requisitos de interface e usabilidade 6](#_Toc29515779)

[Tabela 3 - Descrição dos atores 9](#_Toc29515780)

[Tabela 4 - Caso de Utilização #1 – Estatísticas de Tráfego 10](#_Toc29515781)

[Tabela 5 - Caso de Utilização #2 - Configuração Sistema 11](#_Toc29515782)

[Tabela 6 - Caso de Utilização #3 - Consultar Informações do Tráfego 12](#_Toc29515783)

[Tabela 7 - Caso de Utilização #4 - Autenticação 12](#_Toc29515784)

[Tabela 8 – Cobertura dos requisitos 13](#_Toc29515785)

# Introdução

No âmbito do Projeto Temático em Desenvolvimento de Aplicações, foi proposto ao grupo conceber uma aplicação, cujo tema escolhido foi “Gestão de Tráfego por Radares”. Este tema consiste na utilização dos dados fornecidos por radares para gerar estatísticas relevantes aos condutores e outras organizações interessadas em estudar o tráfego na zona.

Os dados serão fornecidos ao grupo pelo gestor dos radares da Barra e Costa Nova, que, por sua vez, são colocados numa base de dados criada pelo grupo. A aplicação irá utilizar estes dados para gerar e mostrar estatísticas, tais como velocidade média e contagem de veículos. Com estes dados, os utilizadores da app poderão determinar se existe trânsito na zona ou não.

Estas estatísticas também poderão ser partilhadas com mais detalhe para certas organizações, através de uma interface de login, sendo que as credenciais são fornecidas pelo administrador do sistema.

Estando este projeto incluído no módulo Temático em Desenvolvimento de Aplicações, juntamente com as disciplinas de Engenharia de Software e de Sistemas de Bases de Dados, tinha-se também, como objetivo, aplicar os conhecimentos adquiridos nestas aulas.

Ao longo deste relatório será então descrito todo o processo de desenvolvimento deste projeto, desde o seu planeamento, definição dos requisitos, planificação de tarefas, até à implementação da aplicação.

Para finalizar, será feita uma análise dos resultados, uma pequena reflexão crítica e conclusão.

## Visão geral do sistema

Foram colocados vários radares, na Barra e Costa Nova, estes radares detetam os objetos que estão a passar na zona. Identificam os objetos distinguindo se se trata de um carro, de um veículo pesado, de uma bicicleta/mota ou de um peão. Outros objetos ficam na categoria de objetos não identificados, por exemplo, animais, árvores, etc. O radar consegue acompanhar estes objetos durante um certo troço da estrada, fazendo mais que uma medição nesse troço, sendo que quanto mais tempo acompanhar o objeto mais correta será a medição. Para além da sua categoria, de cada objeto são guardadas mais informações, nomeadamente, a data, hora e zona (do radar) da medição, o sentido e a que velocidade vai. Assim, sabendo quantos carros é que estão naquela zona e a que velocidade vão, percebesse se existe congestionamento na zona.

Os radares após captarem estes dados guardam-nos num ficheiro de texto. O grupo vai trabalhar com os dados a partir destes ficheiros, com o objetivo de criar uma aplicação que informe os seus utilizadores destas várias estatísticas. O grupo configurará este sistema utilizando comunicação por sockets, com um modelo cliente-servidor. O sistema tem de ser configurado de modo a saber quantos radares existem, onde é que se encontram e quantos sentidos e vias existem na estrada. Os dados dos ficheiros provenientes do radar são colocados numa base de dados própria, que depois será usada pela aplicação, esta tem de ser configurada de modo a fazer os cálculos necessários para mostrar as várias informações num dado intervalo de tempo.

Relativamente à aplicação esta terá dois tipos de utilizadores, os com e os sem registo, assim, tem de se criar uma lista dos utilizadores com os seus respetivos logins e passwords. As informações que cada um dos tipos de utilizadores terão acesso não serão as mesmas, pois as suas necessidades também serão diferentes.

O utilizador sem registo será, por exemplo, um condutor normal que quer ir passear e procura saber se ir para a praia será uma boa ideia naquela hora ou se estará muito trânsito. Assim, ele entra na aplicação rapidamente (sem precisar de fazer login), procura a informação que necessita, que neste caso, é a contagem dos carros e a velocidade média no momento. Sabendo isto, ele depois toma a decisão de ir ou não para a praia.

O utilizador com registo, por exemplo o estado, pretende perceber se aquela zona necessita de melhorias no que toca ao controlo do congestionamento. Eles irão entrar na aplicação, usando credenciais fornecidas pelo administrador, após terem-se registado previamente. De seguida, eles irão consultar o histórico das velocidades. Para perceberem quais as horas e dias em que houve mais movimento e qual a quantidade de carros que circulam na zona. Assim sabem se será necessário tomar certas medidas para melhorar a circulação nestas vias, melhorar a forma de entrada na zona, ou então irão usar estes dados apenas para efeitos estatísticos.

## Cliente

Para este projeto, o cliente é a empresa que contratou o grupo para criar esta aplicação. Esta empresa pretendia que fosse criada uma aplicação que fizesse a gestão do tráfego nas diversas zonas onde estão colocados os seus radares. Deixando a critério do grupo quais as estatísticas a serem geradas, quais os seus requisitos e casos de utilização. O cliente não necessita de ter acesso a como são geradas as estatísticas, apenas precisa de configurar a aplicação para funcionar para cada um dos diferentes radares.

## Objetivos

Assim, tendo em conta a visão geral deste projeto podem-se agrupar os seguintes objetivos:

* Conceber uma aplicação que permita a gestão do tráfego para qualquer zona à escolha do administrador;
* Permitir a qualquer condutor ter a possibilidade de tomar decisões com base na informação do trânsito obtida pela aplicação;
* Permitir a certas entidades o acesso a informações mais detalhadas do trânsito na zona, para estudos estatísticos ou, eventualmente, para melhorar a circulação do trânsito na área.

# Planeamento

Relativamente ao planeamento do projeto, optou-se por seguir um método ágil. Sendo assim, o planeamento foi dividido em várias iterações. Para cada uma das iterações, iam sendo definidas várias tarefas e as suas respetivas durações estimadas. Ou seja, logo no início do projeto, definiram-se as primeiras tarefas da primeira iteração, definindo-se também uma data limite para o cumprimento daquela iteração. Foram feitas iterações de aproximadamente 1 ou 2 semanas, tirando a última que foi de 4 semanas.

O grupo reuniu-se com o orientador quase todas as semanas, nessas reuniões, não só se discutia o que já tinha sido feito anteriormente como também se discutia quais deviam ser os passos seguintes. Ajudando assim a perceber se a iteração estava a ser bem planeada e no que consistiria a próxima. Assim, foi obtido um único cronograma que ia sendo ajustado, ao longo do projeto iam sendo acrescentadas as várias iterações planeadas. Este cronograma pode ser consultado no Anexo A – Planeamento, do relatório e também no ficheiro Excel entregado em anexo, este ficheiro possui também uma tabela com todas as tarefas definidas.

Nas primeiras fases, durante o planeamento, definem-se os objetivos do trabalho, faz-se o levantamento dos casos de utilização e dos requisitos da aplicação. Elaboram-se vários diagramas. Começando-se pelo diagrama de casos de utilização, o diagrama de classes, com os seus respetivos atributos e métodos mais relevantes e o diagrama de entidades, o diagrama físico da base de dados. Usando um método ágil, a implementação do projeto deve ser dirigida por testes, assim feito o planeamento passa-se à criação dos casos de teste. Ao criarem-se os casos de teste, definem-se os métodos, isto é, define-se o que cada método deve fazer, quais devem ser os resultados esperados tendo em conta os parâmetros introduzidos. De seguida, com base no que se definiu, procede-se à implementação e programação das várias classes, métodos, interfaces e criação da base de dados. Tendo isto em consideração, o grupo tentou seguir todos estes passos de um método ágil, passando por todas as suas fases.

Decidiu-se ainda que a partilha da documentação iria ser feita usando as ferramentas “Git” e “GitHub”. O “Git” é um sistema de controle de versões, permite desenvolver projetos com várias pessoas em simultâneo, editando e criando arquivos, sem o risco de suas alterações serem sobrescritas. O “GitHub” usa o “Git” e permite a criação de repositórios públicos ou privados. Com o “GitHub” é possível verificar com facilidade quais as mudanças que foram efetuadas no projeto e ainda quem as fez. Assim, foi criado um repositório para o grupo, todos os membros tinham acesso a este repositório e era nele que se colocava toda a documentação relativa ao projeto.

# Modelo de requisitos

## Requisitos funcionais

Para qualquer projeto é crucial definirem-se as funcionalidades do sistema a desenvolver. Assim, estabelecida a visão geral do sistema e quais os seus objetivos, passou-se para a definição dos requisitos, começando então pelos requisitos funcionais.

A seguinte tabela, tabela 1, trata-se de um inventário das funções que o sistema tem de suportar e das suas prioridades.

**Tabela 1 - Requisitos Funcionais**

|  |  |  |
| --- | --- | --- |
| Refª | Requisito funcional | Prioridade |
| RF.1 | Permite a consulta das informações relativas às velocidades das viaturas em ambos os sentidos (histórico de velocidades). | +++ |
| RF.2 | Permite a consulta da contagem das viaturas que passam pelo radar (tráfego). | +++ |
| RF.3 | Permite a consulta da velocidade média em intervalos de 10 em 10 minutos. | +++ |
| RF.4 | Sistema indica a velocidade máxima e mínima. | ++ |
| RF.5 | Os condutores (sem registo) só terão acesso à velocidade média de trânsito (de 10 em 10 min) e ao contador de viaturas. | +++ |
| RF.6 | As entidades registadas (Governo, IMT, Brisa…), terão acesso a todas as estatísticas disponíveis. | +++ |
| RF.7 | Permitir a configuração da aplicação, de modo a funcionar para qualquer radar, independentemente da sua localização. | - |
| RF.8 | Permitir o registo e eliminação de entidades. | ++ |

## Restrições e requisitos não funcionais

### Requisitos de interface e facilidade de uso

Definidos os requisitos funcionais, passaram-se para não funcionais. Assim, na tabela abaixo, tabela 2, estão apresentados os requisitos que não são essenciais para o funcionamento do serviço, mas que facilitam a navegação da interface para o utilizador.

**Tabela 2 - Requisitos de interface e usabilidade**

|  |  |
| --- | --- |
| Refª | Requisito de interface e usabilidade |
| RInt.1 | Dados têm de se atualizar de 10 em 10 minutos. |
| RInt.2 | A interface do condutor é a interface principal, sendo logo mostrada quando se abre a aplicação. |
| RInt.3 | A interface do condutor possui um mapa com a localização do radar, tendo também indicação do número de vias e sentidos nessa zona. |
| RInt.4 | Mapa tem de ser manipulável, mas será possível afastar-se muito da zona do radar. |
| RInt.5 | A interface do condutor possui também uma tabela com as informações da velocidade média para cada um dos sentidos, a informação da contagem dos carros, mostrando apenas a diferença e a avaliação do trânsito (muito trânsito, trânsito moderado e pouco trânsito). |
| RInt.6 | A interface das entidades possui uma tabela com as velocidades máxima e mínima e um gráfico com as velocidades médias de 10 em 10 minutos. Pode também ser aberto noutra página uma tabela com um histórico de todas as velocidades. |
| RInt.7 | O administrador tem ainda acesso a uma interface de configuração onde faz o registo das novas entidades e as outras configurações da aplicação. |

### Requisitos de desempenho

Requisitos e restrições que não são essenciais para o serviço, mas que diminuem a carga de processamento do servidor, evitando situações que atrasam a resposta de servidor

|  |  |
| --- | --- |
| Refª | Requisito de desempenho |
| RDes.1 | Limite máximo de 254 clientes ligados ao mesmo tempo |
| RDes.2 | Existir um servidor adicional caso o primário ir abaixo |

### Requisitos de segurança e integridade dos dados

[relacionar requisitos de segurança com os requisitos funcionais identificados, quando aplicável]

|  |  |
| --- | --- |
| Refª | Requisito de segurança, privacidade e integridade de dados |
| RSeg.1 | Não são guardados qualquer tipo de dados dos condutores. |
| RSeg.2 | Os dados das entidades são privados e não podem ser partilhados. |

# Modelo de Casos de Utilização

## Visão geral

O primeiro diagrama que se deve fazer é o diagrama de casos de utilização. Para criar este diagrama é necessário saber os cenários de utilização da aplicação. Estes cenários já foram definidos previamente e foi a partir deles que se criou o seguinte diagrama apresentado na figura 1.

Como já foi referido, esta aplicação iria a partir dos dados recebidos pelo radar gerar estatísticas que iram depois ser consultadas pelos condutores, sendo que estes podem estar ou não registados. A aplicação teria um administrador encarregue da sua configuração. Assim, os casos de uso mais importantes são a criação das estatísticas, a consulta das mesmas, a autenticação e a configuração do sistema.

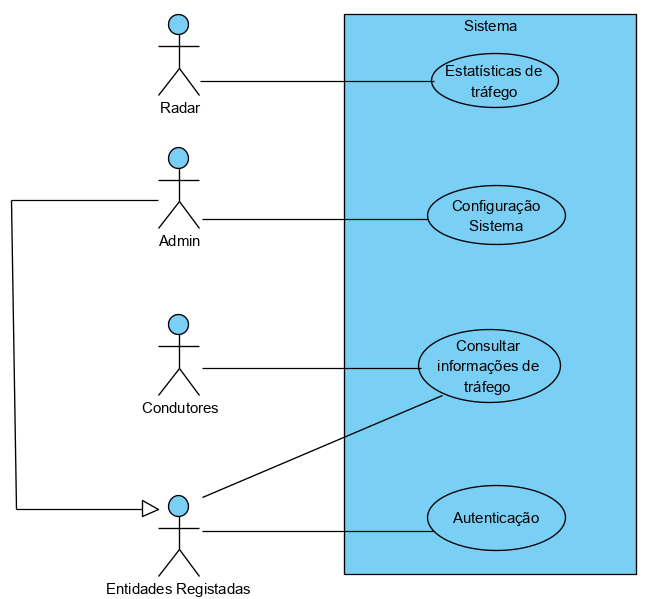


Figura - Diagrama de Casos de Utilização

## Atores

A construção de um diagrama de casos de utilização envolve também a definição dos atores que estão envolvidos com o sistema. Cada um deles interage com a aplicação/sistema através de pelo menos um dos casos de utilização representados. Estes atores estão descritos na tabela 3.

**Tabela 3 - Descrição dos atores**

| Ator | Descrição |
| --- | --- |
| **Radar** | Equipamento que recolhe dados brutos do tráfego na zona. |
| **Administrador** | Indivíduo que configura o sistema e faz a sua manutenção, especialização das Entidades Registadas, herdando as suas características. |
| **Condutores** | Utilizadores normais que pretendem fazer consultas simples de tráfego. |
| **Entidades Registadas** | Utilizadores com login que têm acesso a informações mais detalhadas. |

## Descrição dos casos de utilização

De seguida, para uma melhor e mais detalhada explicação do diagrama de casos de utilização e do que este representa, passar-se-á para uma descrição de cada caso de utilização. Sendo que, para os casos em que seja pertinente, isto é, para os casos mais complexos, serão apresentados diagramas de atividade. Os diagramas de atividades são fluxogramas que decompões uma tarefa/atividade em subactividades, ajudando assim a explicar no que consiste o caso e quais as interações dos atores com o sistema.

### [Estatísticas de Tráfego #1]

**Tabela 4 - Caso de Utilização #1 – Estatísticas de Tráfego**

| Nome: | [Estatísticas de Tráfego #1] |
| --- | --- |
| Atores: | Radar (inicia) |
| Prioridade: | Alta |
| Finalidade: | Processar os dados do radar para gerar estatísticas de tráfego. |
| Pré-condições: | Receber dados provenientes do Radar. |
| Sumário: | Os dados fornecidos pelo radar serão processados pelo sistema de forma a criar estatísticas úteis. Vão ser utilizadas as velocidades para saber o sentido das viaturas (velocidade negativa e positiva referem-se a sentidos diferentes) e para calcular as velocidades médias de 10 em 10 minutos. É, também, criado um histórico de velocidades e vai ser guardada a velocidade máxima e mínima desde sempre. Vão ser utilizados os IDs dos objetos para fazer a contagem das viaturas de cada um dos sentidos, calculando também a sua diferença, dando a indicação de quantas viaturas estão naquela zona. |

#### Diagrama de Atividades



Figura - Diagrama de Atividades – Estatísticas de Tráfego

### [Configuração Sistema #2]

**Tabela 5 - Caso de Utilização #2 - Configuração Sistema**

| Nome: | [Configuração Sistema #2] |
| --- | --- |
| Atores: | Administrador(inicia) |
| Prioridade: | Baixa |
| Finalidade: | Configuração da aplicação para cada radar, escolhendo que dados vão ser exibidos. |
| Pré-condições: | Possuir aplicação e interfaces criadas e uma interface de configuração. |
| Sumário: | O administrador configura o sistema para cada radar, podendo usar esta aplicação independentemente de qual o radar e de qual a sua localização. Administrador tem de indicar quantos radares existem e onde estão, quantas vias existem para cada um dos sentidos. Feita a definição da localização do radar, configura quais as estatísticas que vão estar disponíveis para cada um dos tipos de utilizadores da aplicação. Terá também uma interface para fazer o registo das entidades. |

#### Diagrama de Atividades



Figura - Diagrama de Atividades – Configuração Sistema

### [Consultar informações do tráfego #3]

**Tabela 6 - Caso de Utilização #3 - Consultar Informações do Tráfego**

| Nome: | [ Consultar informações do tráfego #3] |
| --- | --- |
| Atores: | Condutores e Entidades registadas (inicia) |
| Prioridade: | Alta |
| Finalidade: | Consultar a interface principal da aplicação. |
| Pré-condições: | Interfaces criadas, sistema a gerar estatísticas e configuração do sistema feita. |
| Sumário: | Utilizador abre a aplicação. Sistema mostra a interface principal, que corresponde à interface com as informações para os condutores, utilizadores sem registo, ou seja, são as estatísticas menos detalhadas. No caso dos condutores não farão mais nada, no caso das entidades registadas podem prosseguir à autenticação. |

### [Autenticação #4]

**Tabela 7 - Caso de Utilização #4 - Autenticação**

| Nome: | [Autenticação #4] |
| --- | --- |
| Atores: | Entidades Registadas (inicia) |
| Prioridade: | Alta |
| Finalidade: | Permitir o acesso a outras informações que utilizadores sem login não podem ver. |
| Pré-condições: | Entidades têm de fazer um registo prévio contactando o administrador diretamente. |
| Sumário: | Utilizador abre a aplicação. Sistema mostra a interface principal. Utilizador escolhe a opção de fazer login. Sistema mostra a interface de autenticação. Utilizador insere as credenciais. No caso do administrador, ele é redirecionado para a sua interface própria de configuração, se for uma entidade registada, mostra a interface destes utilizadores, que possui mais informações estatísticas que a do condutor. |

## 

## Cobertura de requisitos

Cada caso de utilização implementa ou pelo menos está relacionado com algum dos requisitos funcionais. Assim, na tabela 8 está representada essa mesma relação dos casos de utilização e os requisitos.

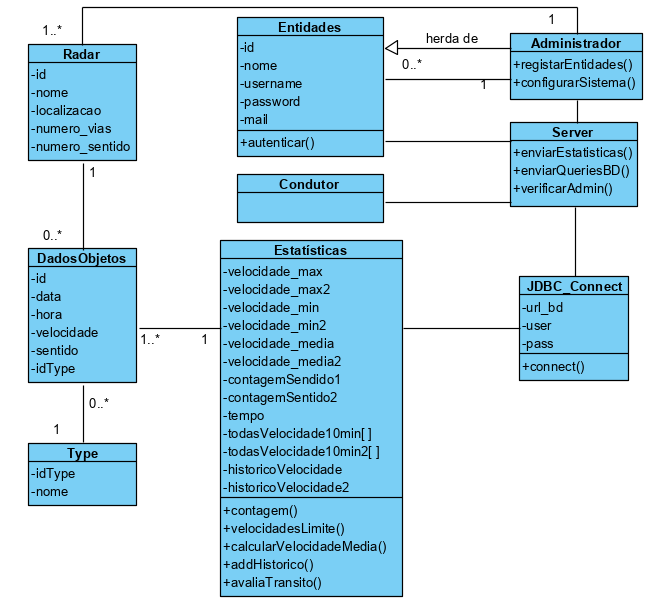
**Tabela 8 – Cobertura dos requisitos**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RF1 | RF2 | RF3 | RF4 | RF5 | RF6 | RF7 | RF8 |
| Estatísticas de Tráfego #1 | X | X | X | X |  |  |  |  |
| Configuração Sistema #2 |  |  |  |  | X | X | X | X |
| Consultar informações do tráfego #3 | X | X | X | X | X |  |  |  |
| Autenticação #4 |  |  |  |  |  | X | X | X |

# Diagrama de Classes

A partir do diagrama de casos de utilização, definiram-se os principais conceitos deste sistema e elaborou-se o diagrama de classes. Para cada classe definiram-se os atributos e métodos que fossem necessários a implementar. A criação deste diagrama demorou algum tempo e até à fase da implementação, este foi ajustado e corrigido várias vezes até ser obtido o diagrama apresentado na figura 4.

Figura - Diagrama de Classes



De seguida, será feita uma pequena descrição de cada uma das classes presentes no diagrama de classes:

* **Radar -** Esta classe representa cada radar, sendo que cada um está associado a um ID. Aqui pode-se encontrar também como atributos, o seu nome e localização, o número de vias e sentidos que ele monitoriza. Cada Radar tem um Administrador e cria vários objetos da classe DadosObjetos.
* **DadosObjetos -** Esta classe representa os objetos lidos pelos radares. Ou seja, cada objeto desta classe é um objeto que passou pela estrada, tendo um ID, um registo da hora, velocidade, sentido e tipo (carro, mota, etc.). Cada objeto desta classe tem um só Radar, um só Type.
* **Type -** Como os radares conseguem distinguir os diferentes tipos de objetos que passam por eles, foi criada esta classe para guardar o nome da classificação dos objetos, associando esse nome a um ID próprio. Cada uma destas classificações pode estar associada a vários objetos da classe DadosObjetos.
* **Estatísticas** - Esta classe representa as estatísticas geradas com base nos vários objetos da classe DadosObjetos. Assim esta classe tem como atributos o tempo, e para cada sentido, um array com todas as velocidades de 10 minutos, a contagem, a velocidade média, máxima e mínima e um histórico com todas as velocidades. Esta classe tem ainda vários métodos associados, que servirão para gerar as estatísticas.
* **Entidades –** Esta classe representa os utilizadores com registo. Tendo como atributos, o id, o nome, *username*, email e password. Sendo esta classe um cliente, está associada à classe Server.
* **Administrador** – É uma especialização da classe Entidades, herdando todos os seus atributos e métodos. Contudo, terá algumas particularidades próprias, como é o caso dos métodos a mais que possui.
* **Condutor -** É uma classe que representa os utilizadores não registados, sendo também um cliente estará associada à classe Server. Esta classe não tem atributos pois deste utilizador não será guardada nenhuma informação.
* **Server -** Esta classe que representa o servidor da aplicação, estando associada às classes que representam os utilizadores da aplicação.
* **JDBC\_Connect -** Esta classe servirá para estabelecer a conexão entre o servidor e a base de dados, tendo como atributos o url\_bd, o user e a pass da base de dados.

# Diagrama de Entidades

Para a criação do diagrama de entidades, primeiro analisou-se o diagrama de classes de modo a perceber-se quais as classes que iam necessitar de estar presentes na base de dados. Teve-se ainda de perceber se eram precisas mais classes, de modo a garantir que a base de dados cumpria as regras e que esta era criada corretamente.

O diagrama de entidades também foi sendo ajustado, mas conclui-se que o diagrama da base de dados seria o que está apresentado na figura 5.



Figura - Diagrama de Entidades

Como se pode verificar, existem várias tabelas que são comuns ao diagrama de classe, mas existem algumas que foram acrescentadas apenas aqui. De seguida, tentar-se-á esclarecer a razão destas decisões.

As tabelas que são comuns ao diagrama de classes são:

• **Radar** – Esta tabela guarda as informações dos vários radares. Cada Radar possui como chave primária um ID único. Tem ainda, o seu nome e localização como varchar, e o número de vias e sentidos que monitoriza como integer.

•  **DadosObjetos** – Esta tabela guarda todos os dados registados pelos radares. Cada objeto tem como chave primária um ID único. Existindo ainda o registo da hora, como timestamp, a velocidade, como integer, e o sentido como varchar. Tem ainda como chaves estrangeiras, as chaves primárias das tabelas Radar, id\_radar, e Type\_Objeto, id\_type. Uma vez que, cada objeto é lido por um certo radar e pertence a um dos tipos de objetos.

• **Type\_Objeto** – Esta tabela guarda o nome das classificações dos objetos e associa-os a um ID, id\_type, que serve com chave primária da tabela.

• **Entidades** – Nesta tabela estão registadas as informações das entidades registadas. Estas informações incluem o nome, username, email, como varchar, e tipo de entidade (Type\_Entidadeid) e um ID único como integer.

Não foi criada uma tabela para o Administrador, invés disso, acrescentou-se uma tabela:

• **Type\_Entidades** – Aqui pode se encontrar o significado do tipo de entidade registado na tabela de entidades, sendo 0 utilizador administrativo e 1 um utilizador não administrativo. Desta forma a tabela Entidades recebe a chave primária desta tabela, que é o id, como chave estrangeira. Assim, mesmo sem a tabela Administrador, já é possível distinguir facilmente quem são os administradores dentro da tabela das Entidades, o que é útil pois estes terão acesso a mais ferramentas na interface a desenvolver.

Por motivos de segurança, foi também criada uma tabela Login:

• **Login** – Esta tabela guarda o username e password das entidades, sendo o user a chave primária. Desta forma, é possível manter as passwords numa tabela separada da tabela das Entidades, mas existe na mesma uma relação, sendo que as entidades recebem como chave estrangeira o username.

Criou-se ainda uma outra tabela, que iria servir para guardar as definições dos estados de transito, para depois ser feita a avaliação do trânsito:

• **Estado** – Esta tabela associa um certo limite de velocidade a uma descrição da avaliação de trânsito e depois cada classificação é identificada por um id. Por exemplo, se a velocidade for inferior a 20 km/h está muito trânsito.

No diagrama de classes, definiu-se também uma classe chamada Estatísticas, nesta classe, iriam estar todas as estatísticas geradas a partir dos dados recolhidos. Contudo, para a base de dados optou-se por invés de numa tabela, criar estas estatísticas usando vistas materializadas. Uma vista materializada representa o resultado de uma consulta, que será atualizado periodicamente a partir das tabelas originais.

Assim, para gerar as estatísticas necessárias iriam ser criadas vistas materializadas a partir dos dados da tabela DadosObjetos. Foram divididas as estatísticas de cada sentido em vistas diferentes para mais fácil distinção. A seguir serão descritos os motivos de criação das várias vistas. Para simplicidade, só se vai falar de um dos sentidos:

* **histórico\_sentido –** Esta vista serve de histórico para velocidades registados pelo radar, estando cada registo associado a uma *timestamp* (Data e hora) e a velocidade.
* **sentido\_last\_10 –** Nesta vista é guardada a contagem e a velocidade média, máxima e mínima das viaturas que passaram pelo radar nos últimos 10 minutos. Esta tabela atualiza-se a cada 10 minutos.
* **estatisiticas\_sentido -** Esta vista serve para fazer a avaliação do estado do trânsito, utilizando as velocidades médias e contagens registadas na vista **sentido\_last\_10**. Assim, são guardados os mesmos campos da vista sentido\_last\_10 mas também o estado de trânsito correspondente. Esta tabela também se atualiza a cada 10 minutos.

# Implementação da Base de Dados

Após a criação do Diagrama de Entidades (figura 5), prosseguiu-se à implementação da Base de Dados. Para a criação da base de dados utilizou-se o sistema PostgreSQL e a plataforma pgAdmin, dois softwares open source potentes que utilizam principalmente programação SQL para a realização das suas tarefas. Foram escolhidos estes softwares pela sua popularidade e pelas suas facilidades de uso, já que o pgAdmin permite fácil manipulação das tabelas e dos seus dados. O código para a implementação da base de dados poderá ser encontrado em Anexo/FicheiroSQL?.

## Criação das tabelas

No diagrama de entidades definiram-se as seguintes tabelas: **Radar, DadosObjetos, Type\_Objeto, Entidades, Type\_Entidades, Login** e **Estado.** A criação da base de dados foi feita, naturalmente, a partir do que tinha sido planeado. Assim, iriam ser implementadas todas estas tabelas e as relações entre elas, colocando as devidas chaves primárias e estrangeiras.

Contudo, a base de dados acabou por sofrer algumas alterações, nomeadamente pelo facto de não se ter criado a tabela Estado. Esta tabela iria servir para guardar as definições da avaliação de trânsito, porém esta funcionalidade acabou por apenas ser implementada nas vistas materializadas. Assim, como a tabela não estava a ser usada, foi retirada da base de dados. Fez-se ainda uma alteração na tabela DadosObjetos, passando o campo sentido de varchar para integer.

## Criação das views

Todas as vistas materializadas que tinham sido definidas foram criadas na base de dados. Contudo, como o grupo não teve acesso a uma *live* *datastream* dos dados de um radar, não estariam sempre a ser recebidos dados na base de dados. Desta forma, as vistas que geram as estatísticas dos últimos 10 minutos, nunca gerariam nada. Teve então de se criar vistas que gerassem estatísticas, para que se pudesse testar o código na aplicação, utilizando unicamente ao ficheiro de excel fornecido. Assim, foram criadas as seguintes vistas para cada sentido:

* **sentido\_all –** Esta vista equivale à vista sentido\_last\_10, mas invés de ser apenas com os dados dos últimos 10 minutos, são usados os dados do último ano.
* **sentido –** Esta vista equivale à vista estatisiticas\_sentido, mas invés de usar a vista sentido\_last\_10, é criada a partir da vista sentido\_all.

## Filtração e inserção dos dados do radar na base de dados

Para se poder testar a aplicação, foi fornecido um ficheiro Excel pelo orientador do grupo, que possuía os registos obtidos por um radar na Costa Nova num intervalo de 4 horas.

Este ficheiro foi analisado pelo grupo e alterado de forma a ser facilmente importado para a tabela **dadosobjetos**. Foram retiradas do ficheiro as colunas de ‘*UMRRID*’, ‘*BusNo*’, ‘*MilliSecs*’, ‘*Seconds’*, ‘*Speed* *[m/s]*’ e ‘*L’* pois foram considerados informações desnecessários, já que não iriam ser utilizados nas tabelas nem mostrados na interface da aplicação. Após isto, foram retirados os registos cujos valores não fossem ‘0’ das colunas ‘*ML/(Sensor)Zone’* e ‘*MesaZone’* para que não houvesse registos duplicados de objetos, já que o mesmo objeto poderia ser detetado em duas zonas diferentes ao mesmo tempo.

Como os ‘*ObjectID’s* repetiam a cada 255 registos, foram renumerados de forma a que isto não acontecesse. Foi ainda adicionada uma coluna que distinguisse o sentido do objeto, baseado no sinal das velocidades associados, sendo ‘0’ associado a velocidades negativos, e ‘1’ associado a velocidades positivos. Por último, adicionou-se uma coluna ‘id\_radar’, existindo nesta coluna unicamente o valor ‘1’, já que todos os registos eram do mesmo radar.

Após a filtração dos dados, o ficheiro foi exportado para formato CSV e importado na tabela **dadosobjetos** através do pgAdmin.

# Métodos e Casos de Teste

## Principais métodos a implementar

Para se poder proceder à implementação e codificação da aplicação, primeiro tinham de se definir quais os principais métodos a ser implementados. Desta forma, o grupo conseguia ter uma noção do que é que teria de implementar:

* **enviarEstatisticas() -** Este método vai servir para mostrar todas as estatísticas aos clientes. Tem de enviar ao cliente todos os dados do radar, mostrar o histórico, mostrar as estatísticas de cada um dos sentidos. Vai à base de dados e retira de lá os dados, enviando-os ao cliente através da conexão Socket.
* **autenticarUser(**String user, String pass**) -** Este método tem de receber o user e a pass, para depois verificar se existe na bd. Se existir e coincidir retorna *true*, se não retorna *false*.
* **verificarAdmin(**String user, String tipo**)** - Este método tem de receber o user e o tipo de entidade, campo da Type\_Entidade da tabela Entidades na base de dados. Depois vai verificar se é um administrador ou não. Retorna true se for administrador e false se não for.
* **registarEntidades(**String nome, String user, String mail, String type**) -** Este método fará o registo de novas entidades, inserindo-as na Base de Dados. Recebe os dados da entidade a registar. Cria uma pass gerada automaticamente. Coloca o user e a pass na tabela login da Base de Dados e coloca o nome, user, mail e tipo na tabela das entidades da BD. Retorna true se conseguir efetuar o registo.
* **exQuery(**String query**)** - Este método recebe uma string que será a query, envia-a à base de dados e não irá retornar nada
* **connect() –** Este método irá fazer a conexão com a BD, não tem parâmetros, mas vai retornar a ligação com a base de dados
* **configurarSistema() –** Este método iria servir para o administrador fazer a configuração do sistema, adicionar novos radares, etc. Contudo, esta funcionalidade era menos prioritária então o grupo decidiu-se focar antes nos outros métodos.

## Casos de Teste

# Testes Unitários

Testes de Unidade ou teste unitário é a fase de testes onde cada unidade do sistema é testada individualmente. Unit, é o nome genérico para qualquer estrutura de testes automáticos unitários. Consiste na verificação da menor unidade do projeto de software.

O objetivo é isolar cada parte do sistema para garantir que elas estejam funcionando conforme especificado. Através da utilização deste tipo de teste ao longo da implementação é possível reduzir a quantidade de bugs na aplicação final. Os testes do Unit funcionam através de comparação de resultados das funções a serem testadas com valores esperados.

Neste caso não seria apenas necessário criar testes só com o JUnit pois após algumas pesquisas na internet foi descoberto que como para a classe “JDBCConnect” estávamos a fazer uma conexão à base de dados, na altura de fazer os testes dava um erro, pois como a base de dados está sempre em atualização não é possível fazer os testes porque derá sempre erro, foi encontrado um solução “JMock”.

JMock é uma biblioteca que auxilia o Test Driven Development (TDD) através dos mock objects. É uma biblioteca que vai criar implementações falsas específicas para os testes que queremos testar. Mas para a utilização do JMock é necessário criar uma pasta Maven para criar a biblioteca necessária para ter as funções necessárias todas. Após muitos testes o grupo não conseguiu usar o JMock devido ao projeto Maven, decidimos procurar outra forma tais como “HSQLDB”, “DBUnit”, “NBUnit” mas todos sem sucesso.

# Implementação da aplicação no NetBeans

Para a implementação das classes e métodos, escolheu-se o NetBeans. O NetBeans IDE é um ambiente de desenvolvimento integrado gratuito e *open source* para programação usando Java. Escolheu-se este programa uma vez que é fácil e simples de usar. Além disso, como já tinha, anteriormente, sido usado nas aulas, o grupo já estava habituado a trabalhar neste programa, não sendo necessário aprender a usar um novo software.

## Arquitetura da aplicação Cliente – Servidor

Neste projeto utilizou-se a arquitetura cliente-servidor, onde se criou uma aplicação que funciona como servidor, permitindo assim processar e enviar vários dados que são pedidos pelos clientes. A aplicação cliente é a mesma para todos os clientes e permite obter diversos dados do servidor. Implementou-se esta arquitetura usando Multithreading de forma a permitir que várias instâncias do cliente corram e recebam dados do servidor em simultâneo.

Quando a aplicação do servidor é aberta, as conexões são criadas e este fica à espera de aceitar clientes, utilizando comunicação por sockets. Além da conexão principal que serve para enviar os dados para o cliente, criou-se também uma conexão adicional que serve para enviar dados críticos para a autenticação. Ou seja, foi criado um servidor para o Login, sendo que este é iniciado a partir da aplicação do servidor principal.

Quando a aplicação do cliente é iniciada, a mesma é automaticamente conectada à socket do servidor. Este vai buscar todos os dados à base de dados e envia-os ao cliente, que os recebe de imediato e mostra-os na sua interface principal. Caso o cliente queira fazer login, ao abrir a janela de login do cliente, essa janela vai ser conectada à socket do servidor dedicado à autenticação. Uma vez feita a autenticação, o cliente terá acesso a funcionalidades adicionais, sendo que se for um administrador terá ainda mais particularidades.

Assim, para explicar melhor a estrutura deste projeto, foi feito um diagrama de componentes, apresentado na figura 6. Este diagrama captura a estrutura física da implementação e é construído como parte da especificação da arquitetura. Contém os componentes, interfaces e ficheiros e as relações entre eles.

## Conexão à base de dados – JDBC

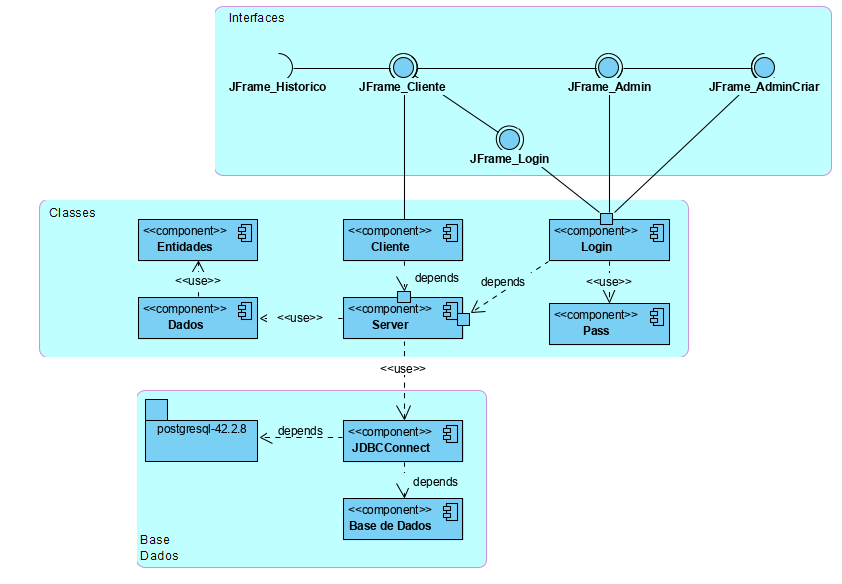


Figura - Diagrama de Componentes

Java Database Connectivity ou JDBC é uma interface de programação de aplicativos (API) que reúne conjuntos de classes e interfaces escritas na linguagem Java que executa instruções SQL de qualquer tipo de banco de dados relacional, define como um cliente pode ter acesso a esse mesmo banco de dados. É uma tecnologia de acesso usada para conectividade de banco de dados Java.

Para este acesso criamos então uma classe chamada JDBCConnect, que faz a conexão de a base de dados com o PgAdmin. Com isto, foram precisas as credenciais do PgAdmin que permitisse o acesso sendo então, as que estão na figura ….

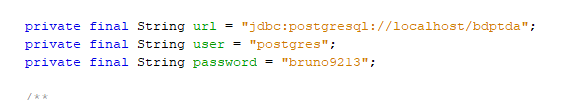


Figura 7- Credenciais

Após termos as credenciais foi feita um método chamado “Connect” que verificaria se as credenciais estariam bem e caso isto se sucede-se retornava ????

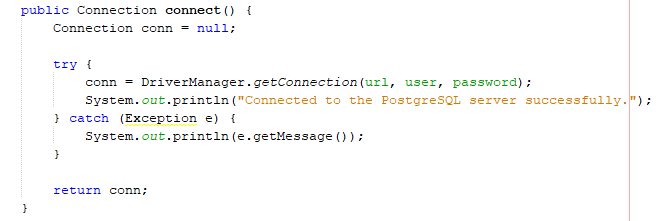


Figura 8- metodo de conectar

O próximo método é utilizado pela nossa aplicação, cada vez que é feita uma chamada à base de dados, isto serve para quando precisamos de algo este método faça o código no PgAdimin de modo a nos garantir o que é suposto ser visto ou obtido, retornando o resultado da query.

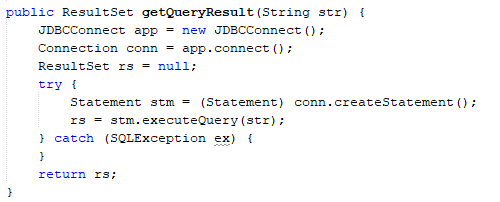


Figura 9- chamada à base de dados

Este método serve para inserir o login na nossa aplicação, sendo que, os usuários têm que ter um registo deste tipo, que contém um “user” e uma “pass”, neste caso irão colocá-la e só será válida se se verificarem corretas e existentes.

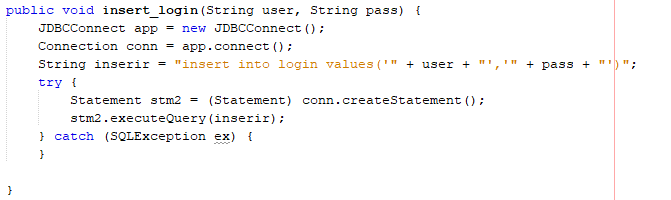


Figura 10-inseir login

Neste caso “insert\_entidades”, é para a criação de uma nova entidade, contendo o seu “id”, “nome”, “user”, “mail” e o “id\_type” que dirá se é um “administrador” ou um “não administrador”.

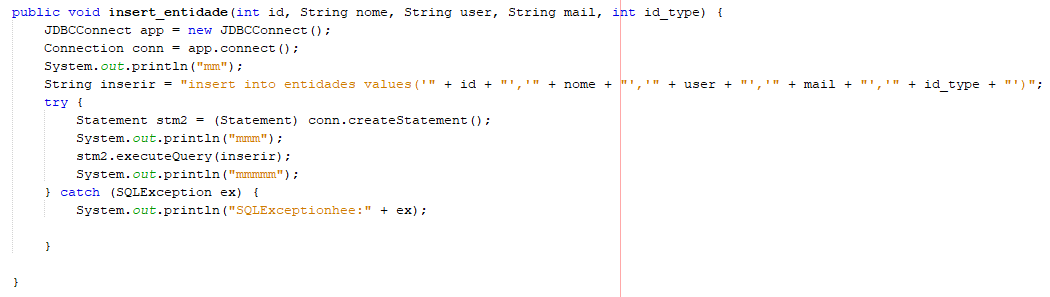


Figura 11-inserir entidades

## Criação interfaces em JFrame

Interfaces

## Implementação classes e métodos

As classes que foram realmente implementadas no NetBeans, acabaram por ser diferentes do que se tinha originalmente planeado. Algumas das classes não foram implementadas no NetBeans porque eram apenas classes de dados, ou seja, seriam implementadas na base de dados, mas não tinham necessariamente de ser criadas como classes java. Como foi o caso da classe Radar, a classe DadosObjetos e a classe Type. Estas foram apenas classes implementadas na base de dados. A classe Estatísticas, que serviria para gerar as várias estatísticas foi apenas implementada através de views na base de dados, tal como todos os seus métodos. Contudo, era na mesma necessário armazenar nalguma classe todos estes dados, de modo a poderem ser enviados para os clientes. Como tal, foi criada a classe **Dados**, onde eram guardados todos os dados que o servidor ia pedir à base de dados. Implementou-se também a classe **Entidades,** para armazenar os dados de todas as entidades para depois serem enviados ao cliente.

Relativamente às classes Administrador e Condutor, estas não foram implementadas. Existindo apenas uma classe **Client,** que seriam todos os utilizadores da aplicação. Acrescentou-se ainda a classe **Login,** para ser feita a autenticação dos clientes e a class **Pass**, para gerar passwords aleatórias. A classe **Server** e a classe **JDBCConnect** foram mantidas, sendo que apenas lhes foram implementados mais métodos.

Assim, de seguida na figura 7, pode-se verificar o diagrama de classes final deste projeto.

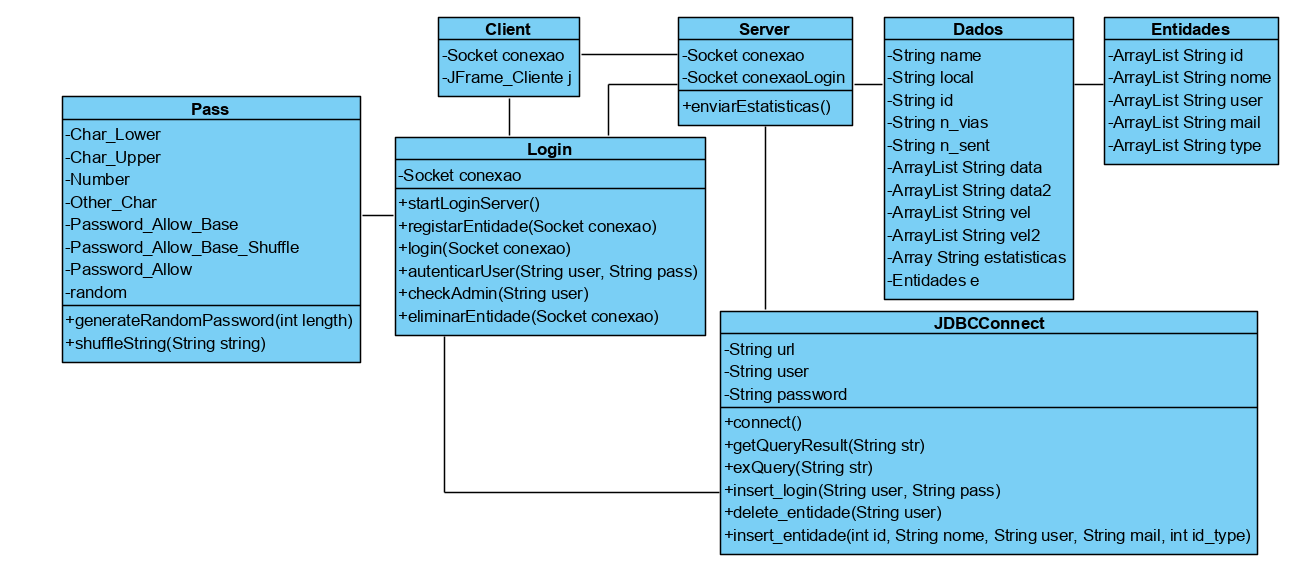


Figura - Diagrama de Classes Real

### Mostrar estatísticas simples

### Mostrar histórico

### Autenticação

### Registo de Entidades

A classe Server fica à espera dos clientes. Assim que um cliente se conecta, o servidor através do método, enviarEstatísticas, vai à base de dados (através da classe JDBCConnect), guarda todos os dados numa instância da classe Dados e envia esse objeto ao cliente. De seguida tendo os dados, as devidas páginas das interfaces iram mostrar os dados.

As entidades são os clientes registados da aplicação. São divididas em 2 tipos administrador e não administrador. A diferença entre elas é que a entidade de tipo administrador pode registar outras entidades. Para registar uma nova entidade é necessário um username, nome e e-mail não registados e validos, caso registados ou inválidos a aplicação avisará o que se passará. A password que é usada no login é criada automaticamente pelo servidor.

Para o Login é necessário um username e uma password validos, caso não sejam a aplicação não procederá e avisa que algo está errado.

O histórico é disponibilizado a todos os utilizadores registados. Utilizadores não registados não têm acesso a essa informação. No histórico podemos verificar todos os registos de velocidade guardados na base de dados e também, a velocidade mínima e máxima. Os registos estão filtrados por sentidos, tendo um botão que nos permite mudar o sentido e assim mudar os dados

Fluxos das interfaces mockups?

Mudanças dos diagramas – análise de resultados

Falar do planeamento – conclusão maybe

Falhas - ???

Testes do sistema – user testing??

# Testes ao sistema ???

# Análise dos Resultados

REQUISITOS CUMPRIDOS E NÃO CUMPRIDOS

OBJETIVOS CUMPRIDOS E NÃO CUMPRIDOS

# Reflexão Crítica e Conclusão

O planeamento do projeto levou mais tempo do que deveria, o que atrasou um pouco o resto do projeto.

No geral, foram cumpridos a maioria dos objetivos e dos requisitos definidos inicialmente, apesar de certas funcionalidades que tinham sido inicialmente planeadas terem sido descartadas, com o intuito do grupo se focar no que realmente era importante. Definiu-se o que tinha a maior prioridade e foi isso que foi implementado.

Em relação ao planeamento do projeto, como se optou por uma abordagem ágil, o desenvolvimento do projeto deveria ser dirigido por testes. Contudo, durante as fases iniciais, o grupo teve algumas dificuldades no planeamento, o que gerou alguma confusão para os membros do grupo. Assim, como o grupo demorou um pouco mais do que devia a perceber o que tinha de ser implementado e como, acabou também por sofrer um pequeno atraso no desenvolvimento do projeto. Mesmo assim, o grupo conseguiu ultrapassar essas dificuldades, definir prioridades, refinar os diagramas e seguir em frente.

Quando se chegou à fase da criação dos casos de testes, estes não foram bem definidos, não sendo logo feitos no JUnit, isso só foi feito posteriormente. Definiram-se quais os principais métodos que tinham de ser implementados no NetBeans, quais os seus parâmetros e o que tinham de retornar. Essa definição foi o suficiente para dar ao grupo uma noção do que tinha de ser implementado, passando então para a parte da programação.

Apesar da criação da Base de Dados não ter gerado muitos problemas, o código foi na mesma refactorizado várias vezes, pois iam sendo encontrados alguns problemas na criação das tabelas e das views.

A implementação da aplicação no NetBeans, foi definitivamente das partes mais complicadas do projeto. A implementação das várias classes foi sendo feita por partes e demorou algum tempo até tudo estar em ordem.

No que toca aos objetivos deste projeto, planeava-se conceber uma aplicação que:

1. permitisse a gestão do tráfego para qualquer zona à escolha do administrador;
2. permitisse a qualquer condutor obter estatísticas do tráfego;
3. permitisse a certas entidades o acesso a informações mais detalhadas do trânsito.

Os objetivos 2 e 3 eram os que tinha prioridade e foram cumpridos. Relativamente ao objetivo 1, este não foi cumprido, uma vez que esta aplicação só permite a gestão de apenas um radar e o administrador não consegue configurar o sistema adicionando novos radares.

Além disso, simplificaram-se as interfaces do utilizador, inicialmente pensou-se em ter um mapa que seria manipulável, mas optou-se pela criação de interfaces mais simples apenas mostrando as estatísticas.

dkflklaflkald

Em conclusão, o projeto merece pelo menos um 15, menos de que isto e a Maria passa-se

# Fontes e material de referência

* Fazer isto arranjar sites e referencias

# Anexos

# Anexo A – Planeamento:

