Projeto de Sistemas de Informação

Enunciado do Projeto — 2024/2025

1 Objetivo

Pretende-se desenvolver uma aplicação *web* para modernizar a gestão de uma empresa de táxis, utilizando as tecnologias MEAN, ou seja, com a base de dados Mongo, as *frameworks* Express e Angular para o desenvolvimento, respetivamente, do *backend* e do *frontend* da aplicação, e o Node.js como ambiente de execução de código JavaScript.

O desenvolvimento da aplicação *web* deve ser inspirado no modelo de processo de *software* Scrum, com três *sprints* de duas semanas cada.[1] Em cada *sprint* pretende-se que seja implementada uma parte das *user stories* da aplicação *web*, havendo uma demonstração logo após o final de cada *sprint*.

Neste documento é inicialmente apresentada uma análise de requisitos bem como um desenho conceptual da base de dados, a partir dos quais se pede que seja feito o desenho lógico da base de dados e sejam implementadas *user stories*. No final do documento, são descritos requisitos adicionais e recomendações, assim como detalhes relativos às entregas de relatórios e avaliações do projeto, bem como um apêndice com aspetos técnicos.

2 Análise de requisitos

Após várias sessões de elicitação de requisitos com os gerentes da empresa de táxis, foram identificadas as seguintes necessidades de informação.[2]

Filiais, táxis, motoristas, e turnos. A empresa tem filiais espalhadas pelo país, cada uma com a sua frota de táxis e os seus motoristas. Cada filial tem um número de identificação de pessoa coletiva (NIPC), um nome (único), um número de telefone, e uma morada, composta por rua, número de porta, código postal, e localidade. Sobre os táxis, guarda-se a matrícula, o ano de compra, a marca, o modelo, e o nível de conforto, que pode ser básico ou luxuoso.

Um motorista pode conduzir qualquer dos táxis da filial à qual está afeto, mas primeiro tem de registar a hora e dia de início e fim de um turno, que pode ser, por exemplo, da noite de um dia até à madrugada do dia seguinte, e só depois escolhe um dos táxis disponíveis, ficando o táxi reservado para o motorista durante o período do turno. As caraterísticas do turno e do táxi são usadas

[1] O guia do Scrum está disponível em https://scrumguides.org/scrum-guide.html.

[2] Estas necessidades de informação são as mesmas do enunciado da etapa 1 do projeto do seu ano letivo da unidade curricular de Sistemas de Informação e Bases de Dados.

para calcular o preço por minuto a cobrar pelo serviço de transporte, podendo ser, por exemplo, mais caro se for feriado ou se o táxi for luxuoso. Um motorista pode conduzir o mesmo táxi mais do que uma vez, desde que em períodos que não se intersetem.

Os dados necessários sobre os motoristas são o número de identificação fiscal (NIF), o nome, o género, o ano de nascimento, a morada onde vive, e o número da carta de condução.

Clientes, viagens, e faturas. Durante um turno de um motorista, cada viagem com clientes é identificada por um número sequencial, começando do um, e são também registadas a morada e hora a que entraram e saíram do táxi (por exemplo, entrada no Aeroporto às 12h de um dia e saída no Campo Grande às 12h30 do mesmo dia), o número de pessoas que foram nessa viagem, e os quilómetros percorridos. Pode assumir-se que estes dados só são registados quando uma viagem termina. Naturalmente, o período em que uma viagem acontece tem de estar contido no turno em que o motorista fez essa viagem.

Após uma viagem, é calculado o preço a pagar com base no número de minutos que a mesma demorou, a multiplicar pelo preço por minuto que tinha sido calculado no início do turno do motorista. Se o cliente quiser, pode ser emitida uma fatura cujo número de sequência está dependente da filial (a primeira fatura de qualquer filial tem o número um), sendo também guardada a data, e o NIF, nome e género do cliente. Para efeitos de auditoria, é necessário saber qual a viagem que corresponde a uma fatura que tenha sido emitida.

3 Desenho conceptual da base de dados

Com base nas necessidades de informação, foi feito o desenho conceptual da base de dados, usando a notação do modelo Entidade-Associação,[3] mostrado na Figura 1, com as restrições de integridade adicionais indicadas a seguir.

Restrições sobre regiões no diagrama

1. A filial à qual o motorista de um turno está afeto tem de ser a mesma à qual o táxi desse turno pertence.
2. O período de tempo de uma viagem de um turno tem de estar contido no período do turno.

Restrições sobre hierarquias

3. Motorista AND Cliente COVER Pessoa.
4. Motorista OVERLAPS Cliente.

[3] A notação do modelo Entidade-Associação é a usada no livro principal de Sistemas de Informação e Bases de Dados: Raghu Ramakrishnan e Johannes Gehrke, *Database Management Systems*, McGraw-Hill, 3ª edição, 2003, ISBN 0072465638.

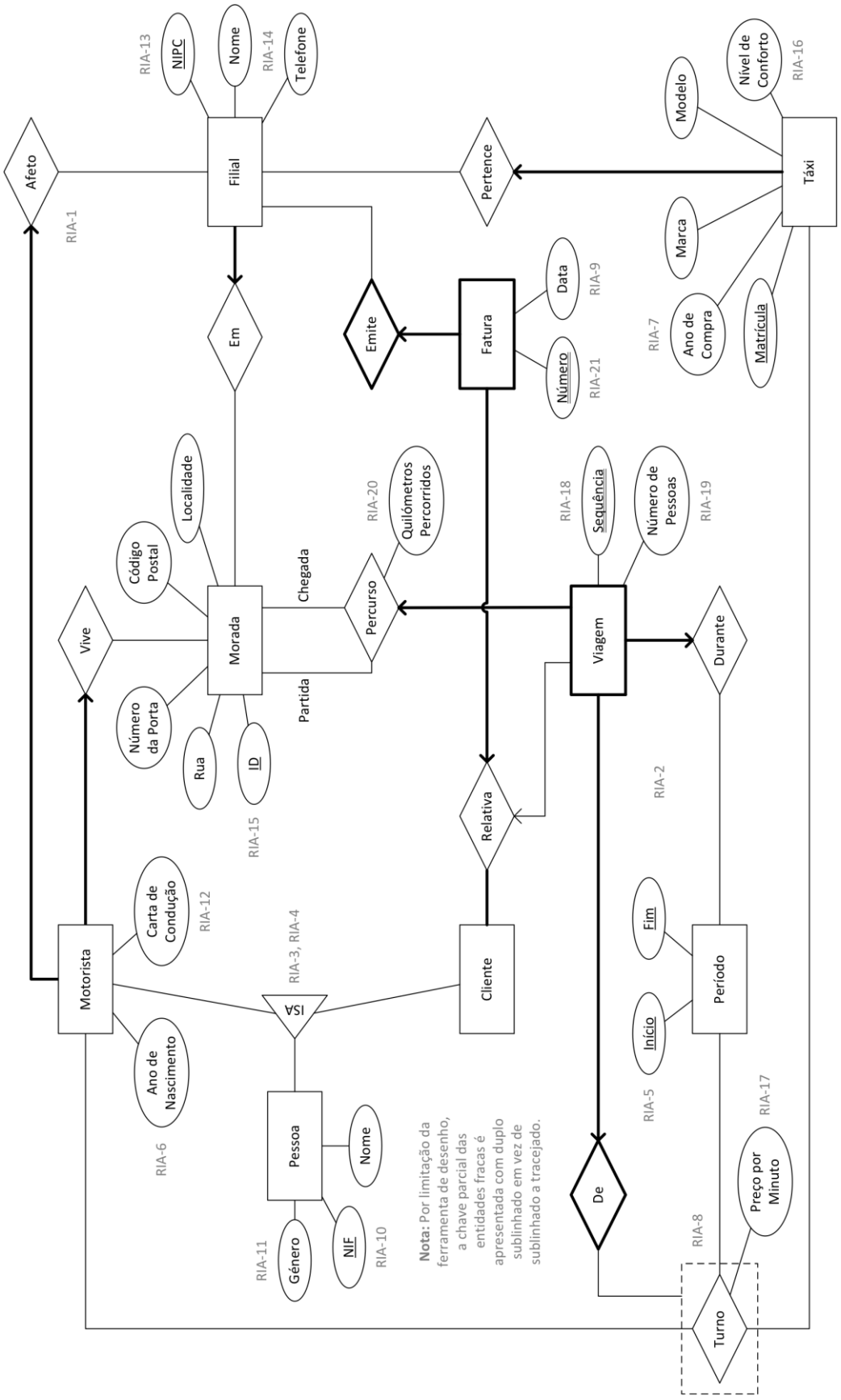

Figura 1: Diagrama Entidade-Associação para a gestão de uma empresa de táxis.

Restrições sobre datas

5. O início de um período tem de ser anterior ao seu fim.
6. O ano de nascimento de um motorista tem de ser 18 ou mais anos anterior ao ano atual.
7. O ano de compra de um táxi tem de ser anterior ou igual ao ano de início do período de qualquer turno no qual participe.
8. Dois turnos do mesmo motorista ou com o mesmo táxi não podem ter períodos que se intersetem.
9. A data da fatura relativa a uma viagem tem de ser posterior à data de início dessa viagem.

Outras restrições

10. O NIF de uma pessoa tem de ter 9 dígitos e ser positivo.
11. O género de uma pessoa tem de ser feminino ou masculino.
12. O número da carta de condução identifica univocamente um motorista.
13. O NIPC e o telefone de uma filial têm de ter 9 dígitos e ser positivos.
14. O nome e o telefone identificam univocamente uma filial.
15. As moradas vão do ID 1 em diante.
16. O nível de conforto de um táxi tem de ser básico ou luxuoso.
17. O preço por minuto num turno tem de ser positivo.
18. As viagens de um turno vão do número de sequência 1 em diante.
19. O número de pessoas que vai numa viagem tem de ser, pelo menos, 1.
20. Os quilómetros percorridos no percurso de uma viagem têm de ser positivos.
21. As faturas emitidas por uma filial vão do número 1 em diante.

Nota importante: a aplicação *web* a desenvolver em Projeto de Sistemas de Informação, sendo um primeiro protótipo, deve *omitir o conceito de filial*, ou seja, os táxis, motoristas, e faturas são diretamente da empresa e não precisam de estar associados a uma filial. Por exemplo, apenas uma fatura terá o número 1, em vez da primeira fatura emitida por cada filial ter o número 1.

4 Desenho lógico da base de dados

A partir do desenho conceptual da base de dados na Figura 1 e das respetivas restrições de integridade adicionais, deve ser elaborado o correspondente desenho lógico, tendo em conta que o sistema de gestão de bases de dados a usar no desenvolvimento da aplicação *web* é o MongoDB.

O manual do MongoDB inclui uma secção sobre modelação de dados[4] e o *blog* da empresa tem um artigo sobre regras práticas para o desenho de esquemas de dados.[5] Também pode ser útil a consulta do guia de migração de bases de dados relacionais para o MongoDB.[6,7]

O desenho lógico da base de dados deve servir de referência para a definição dos modelos de dados a usar no *backend* Express e no *frontend* Angular da aplicação web, incluindo a representação dos dados transferidos em rede.

5 *User stories*

Algumas *user stories* que a aplicação *web* deve suportar decorrem diretamente do texto da análise de requisitos (na página 1), como o registo de um táxi ou de um motorista, e outras ficaram mais bem esclarecidas em reuniões com os gerentes da empresa de táxis. A seguir são enumeradas e descritas as *user stories*, cuja implementação deverá decorrer ao longo de três *sprints*.

Nota: as entidades e restrições de integridade adicionais (RIA) referidas nas *user stories* podem ser encontradas no desenho conceptual da base de dados.

5.1 *Sprint 1*

User Story 1: Como gestor, quero poder **registar um táxi**, para que possa ser conduzido por motoristas em viagens com clientes. Os critérios de aceitação são os seguintes:

a) Deve ser possível preencher um formulário com os dados do táxi, com campos provenientes da entidade Táxi;

b) A matrícula do táxi deve ser validada, não podendo, por exemplo, ter só dígitos ou só letras;[8]

c) A marca e o modelo do táxi devem poder ser selecionados a partir de listas predefinidas;

d) O ano de compra do táxi deve ser anterior ou igual ao ano atual;

e) O nível de conforto do táxi deve satisfazer a RIA 16;

f) Após o registo, o táxi deve aparecer no topo da lista de táxis da empresa, ordenada pela data de criação do registo (mais recente primeiro).

[4] https://www.mongodb.com/docs/manual/data-modeling

[5] https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design

[6] https://www.mongodb.com/resources/solutions/use-cases/rdbms-mongodb-migration-guide

[7] Em Sistemas de Informação e Bases de Dados foi feito o desenho lógico para uma parte da base de dados na Figura 1, em forma de esquema relacional criado em SQL.

[8] Os formatos das matrículas podem ser consultados em https://aran.pt/pt/matriculas-por-ano, interessando apenas as posições possíveis de letras e dígitos, independentemente do ano.

User Story 2: Como gestor, quero poder **registar um motorista**, para que os táxis possam ser conduzidos em viagens com clientes. Os critérios de aceitação são os seguintes:

a) Deve ser possível preencher um formulário com os dados do motorista, com campos provenientes das entidades Motorista, Pessoa, e Morada;[9]

b) O ano de nascimento, NIF, género, e número da carta de condução do motorista devem satisfazer, respetivamente, as RIA 6, 10, 11, e 12;

c) A localidade da morada do motorista deve poder ser automaticamente preenchida a partir de um código postal válido;[10]

d) Após o registo, o motorista deve aparecer no topo da lista de motoristas da empresa, ordenada pela data de criação do registo (mais recente primeiro).

User Story 3: Como gestor, quero poder **definir o preço por minuto do serviço de transporte**, para cobrar as viagens de táxi aos clientes. Os critérios de aceitação são os seguintes:

a) Deve ser possível preencher um formulário com o preço por minuto para cada nível de conforto de táxi, bem como o acréscimo percentual para o período das 21 horas da noite até às 6 horas da manhã (no período diurno não há acréscimo de preço);[11]

b) Os níveis de conforto devem satisfazer a RIA 16 e os respetivos preços por minuto devem ser positivos, conforme a RIA 17;

c) Após a definição do preço por minuto, deve ser possível preencher um formulário para calcular o custo de uma viagem fictícia com início numa hora e fim noutra, eventualmente de dias consecutivos. Por exemplo, se o preço por minuto de um táxi luxuoso for de 0.25€, e houver um agravamento de 20% durante a noite, o custo de uma viagem entre as 20h e as 21h30 é de (60×0.25) + (30×0.25×1.2) = 24.00€.[12]

[9] A RIA 15, referente ao atributo ID da entidade Morada, pode ser ignorada, pois uma morada pode ser identificada pelo *object id* atribuído automaticamente pelo MongoDB. Aliás, as moradas podem nem sequer ser documentos *per se*, nem ter a sua própria coleção.

[10] Os códigos postais e localidades podem ser obtidos em https://www.ctt.pt/ajuda/empresas/apoio-ao-cliente/ferramentas-online/encontrar-codigos-postais (requer o registo de um utilizador) ou em https://github.com/centraldedados/codigos_postais.

[11] Ao contrário do que vem na análise de requisitos e no desenho conceptual da base de dados, ficou esclarecido com os gerentes da empresa de táxis que o preço por minuto não precisa de variar com o dia da semana nem se é feriado ou não.

[12] Uma vez que existe um algoritmo para o cálculo do preço de uma viagem, no desenho conceptual da base de dados, na Figura 1, deixa de fazer sentido a associação Turno ter o atributo Preço por Minuto, interessando registar o Custo Total na entidade Viagem.

5.2 Sprint 2

User Story 4: Como motorista, quero poder **aceder à minha conta**, para poder requisitar táxis e fazer viagens com clientes. Os critérios de aceitação são:

a) Deve ser possível ao motorista digitar manualmente o seu NIF ou escolher a sua conta de utilizador a partir da lista de motoristas da empresa;

b) O NIF do motorista deve satisfazer a RIA 10;

c) Após selecionar a sua conta de utilizador, deve ser mostrada ao motorista uma página onde pode aceder às funcionalidades de requisição de táxi para um turno, aceitação de pedido de táxi, registo de uma viagem com clientes, e emissão de fatura.

User Story 5: Como motorista, quero poder **requisitar um táxi para um turno**, para poder fazer viagens com clientes. Os critérios de aceitação são:

a) Deve ser possível preencher um formulário com os dados do período do turno, com campos provenientes da entidade Período;

b) O início do turno deve ser anterior ao seu fim, conforme a RIA 5, e a duração do turno não deve poder ser superior a oito horas. Adicionalmente, o turno deve começar depois da hora atual e não pode intersetar qualquer outro turno do mesmo motorista, conforme a RIA 8;

c) Após a indicação dos dados do turno, deve aparecer uma lista com os táxis disponíveis para esse turno,[13] de entre os quais o motorista deve poder escolher apenas um;

d) Após a requisição do táxi, deve aparecer uma lista com todos os turnos do motorista e respetivos táxis, ordenada de forma ascendente pela data de início do turno.

User Story 6: Como cliente, quero poder **pedir um táxi**, para chegar mais depressa ao destino. Os critérios de aceitação são os seguintes:

a) Deve ser possível preencher um formulário com os dados do cliente, da sua localização geográfica atual e do destino para onde pretende ir, bem como o nível de conforto do táxi e o número de pessoas que irão no táxi, com campos provenientes das entidades Cliente, Pessoa, Morada, Táxi, e Viagem;

b) O NIF e género do cliente devem satisfazer as RIA 10 e 11, o nível de conforto do táxi deve estar conforme a RIA 16, e o número de pessoas que irá na viagem de táxi deve satisfazer a RIA 19;

[13] Mais precisamente, táxis que não estejam a ser usados em turnos que intersetem o turno em causa, conforme a RIA 8.

c) Se o *browser* tiver acesso às coordenadas geográficas,[14] a morada da localização atual deve ser obtida automaticamente.[15] Caso contrário, a morada deve poder ser preenchida à mão (ex. o código postal) de tal forma que permita a sua correta tradução para coordenadas geográficas;[15]

d) A morada de destino deve poder ser preenchida manualmente, contando que possa ser traduzida para coordenadas geográficas, ou então deve poder ser marcado um ponto num mapa;[16]

e) Após o pedido de táxi, o cliente deve poder ficar a aguardar a resposta de algum motorista, ou eventualmente cancelar o pedido;

f) Se um motorista responder, deve ser mostrado o seu nome, a distância a que está, o tempo estimado de chegada até ao cliente e o custo estimado da viagem até ao destino,[17] e todos os detalhes do táxi. O cliente deve, então, poder aceitar ou rejeitar esse motorista e táxi.

User Story 7: Como motorista, quero poder **aceitar um pedido de táxi**, para poder ir ao encontro de um cliente e transportá-lo numa viagem. Os critérios de aceitação são os seguintes:

a) Deve ser possível visualizar uma lista com os pedidos de táxi ainda não processados, ordenada de forma ascendente pela distância a que os clientes se encontram relativamente à posição atual do motorista.[18] Os pedidos de táxi que não possam ser satisfeitos durante o período do turno atual do motorista[19] não devem ser mostrados na lista, conforme a RIA 2;

[14] https://developer.mozilla.org/ en-US/docs/Web/API/Geolocation_API

[15] Um serviço gratuito de tradução de coordenadas para moradas pode ser encontrado em https://nominatim.org/release-docs/develop/api/Reverse e para fazer a tradução inversa (da morada para coordenadas) pode ser usado https://nominatim.org/release-docs/develop/api/Search.

[16] Uma biblioteca JavaScript para visualização de mapas no *browser* pode ser encontrada em https://leafletjs.com/examples/quick-start. A abordagem de marcar um ponto num mapa para obter uma morada pode também ser adotada na *user story 2*.

[17] Pode ser usada a aproximação de 4 minutos para percorrer 1 km, conforme dados da Carris, em https://lisboaparapessoas.pt/2024/06/24/velocidade-media-autocarros-electricos-carris. Assim, a estimativa do custo de uma viagem pode ser feita aplicando o algoritmo exemplificado na alínea c) da *user story 3* a uma viagem que tem hora de início correspondente à hora atual mais o tempo estimado para o táxi chegar ao local do cliente, e tem hora de fim passado o tempo que demora a percorrer a distância entre o local do cliente e o local de destino.

[18] Para a distância aos clientes poder ser calculada, o *browser* do motorista tem de ter acesso às coordenadas geográficas. Se tal não for possível, deve ser assumida como referência a localização da Faculdade de Ciências: 38.756734, -9.155412. O cálculo da distância entre duas coordenadas pode ser feito com a fórmula de Haversine, disponível em https://rosettacode.org/wiki/Haversine_formula.

[19] Por exemplo, se forem estimados 30 minutos para um motorista ir até um cliente e viajar até ao destino pretendido, e o turno atual desse motorista terminar daqui a 10 minutos, então o pedido desse cliente *não* deve ser mostrado a esse motorista.

b) Cada pedido na lista deve mostrar quantas pessoas querem viajar juntas com o cliente no táxi, em que localização o cliente fez o pedido, a que distância se encontra, e qual o destino pretendido pelo cliente;[20]

c) O motorista deve poder escolher um dos pedidos de táxi e indicar que aceita o pedido, ficando a aguardar a confirmação do cliente. Se o cliente rejeitar ser transportado pelo motorista, esse pedido deve sair da lista.

User Story 8: Como motorista, quero poder **registar uma viagem com clientes**, para poder emitir uma fatura e para poderem ser realizados estudos operacionais e estatísticos. Os critérios de aceitação são os seguintes:

a) A partir do pedido de táxi de um cliente, que foi aceite pelo motorista e confirmado pelo cliente, deve ser possível registar, com o mínimo de introdução manual de dados, que um cliente (e os seus acompanhantes, se existirem) entraram agora no táxi numa dada morada (ver atributos nas entidades Cliente, Período, e Morada associados à entidade Viagem);[21]

b) O número de sequência da viagem no turno do motorista e o número de pessoas no táxi devem satisfazer, respetivamente, as RIA 18 e 19;

c) Quando o táxi chega ao destino do cliente, deve ser possível registar, preferencialmente de forma automática[22] ou com entrada manual de dados, a hora e morada do fim da viagem (ver atributos nas entidades Período e Morada associados à entidade Viagem);

d) O início da viagem deve ser anterior ao seu fim, conforme a RIA 5. Adicionalmente, o período da viagem deve estar contido no período do turno em que a viagem ocorreu, conforme a RIA 2. E ainda, um motorista não deve poder registar viagens cujos períodos se intersetem;

e) A partir das coordenadas geográficas das moradas no início e fim da viagem, deve ser possível calcular automaticamente os quilómetros percorridos,[23] que devem satisfazer a RIA 20;

f) A partir da hora de início e hora de fim da viagem, deve poder ser automaticamente calculado e guardado o preço a pagar pela viagem, que deve poder ser comunicado ao cliente;

g) Após o registo, a viagem deve aparecer no topo da lista de viagens do motorista, ordenada pela data de início da viagem (mais recente primeiro).

[20] Para não sobrecarregar o motorista, a localização atual do cliente e o destino da viagem podem ser apenas nomes de ruas e localidades, com mais detalhes disponíveis, se necessário.

[21] O início e fim de uma viagem podem ser indicados com botões na interface do motorista.

[22] Como a tradução de coordenadas geográficas em moradas, mencionada na *user story 6*.

[23] Ver a fórmula de Haversine referida na nota de rodapé 18.

User Story 9: Como motorista, quero poder **emitir uma fatura referente a uma viagem**, para cumprir a lei. Os critérios de aceitação são os seguintes:

a) Quando uma viagem com um cliente chega ao fim, deve ser possível, com o mínimo de esforço, emitir uma fatura, com campos provenientes das entidades Fatura, Cliente, Pessoa, e Viagem;

b) A data da fatura e o número de exemplar devem satisfazer, respetivamente, as RIA 9 e 21 (esta última adaptada à não existência de filiais);

c) Só deve poder ser emitida uma fatura para cada viagem;

d) Após a emissão, a fatura deve aparecer no topo da lista de faturas emitidas pelo motorista, ordenada de forma descendente pela data da fatura.

5.3 *Sprint 3*

User Story 10: Como gestor, quero poder **editar ou remover um táxi**, para fazer correções aos dados. Os critérios de aceitação são os seguintes:

a) Deve ser possível visualizar uma lista com os táxis existentes na empresa, devendo cada item ter as opções de editar e remover;

b) A opção de remover só deve ser permitida caso o táxi nunca tenha sido requisitado para um turno de um motorista;

c) A opção de editar deve carregar os dados do táxi para um formulário, onde podem ser alterados, aplicando-se, em geral, os critérios de aceitação da *user story 1*;

d) A edição do nível de conforto do táxi só deve ser permitida se o táxi ainda não tiver feito viagens com clientes;

e) Após a edição ou remoção de um táxi, deve voltar a ser mostrada a lista de táxis da empresa, ordenada pela data de atualização do registo (mais recente primeiro).

User Story 11: Como gestor, quero poder **editar ou remover um motorista**, para correções nos dados. Os critérios de aceitação são os seguintes:

a) Deve ser possível visualizar uma lista com os motoristas existentes na empresa, devendo cada item ter as opções de editar e remover;

b) A opção de remover só deve ser permitida caso o motorista não tenha ainda requisitado um táxi para um turno;

c) A opção de editar deve carregar os dados do motorista para um formulário, onde podem ser alterados, aplicando-se, em geral, os critérios de aceitação da *user story 2*;

d) Após a edição ou remoção de um motorista, deve voltar a ser mostrada a lista de motoristas da empresa, ordenada pela data de atualização do registo (mais recente primeiro).

User Story 12: Como gestor, quero poder **visualizar relatórios sobre táxis e motoristas**. Os critérios de aceitação são os seguintes:

a) Deve ser possível visualizar o total de viagens, o total de horas nessas viagens, e o total de quilómetros percorridos nessas viagens, tendo em conta um dado período de tempo, devendo por omissão ser o dia de hoje;

b) Qualquer um dos totais deve poder ser selecionado, para mostrar os subtotais que explicam esse total. Por exemplo, selecionando o total de horas em viagens deve revelar quantas horas fez cada motorista e cada táxi em viagens, com ordenação decrescente (tendo em conta o período de tempo escolhido inicialmente);

c) Qualquer um dos subtotais deve poder ser selecionado, para mostrar os detalhes que explicam esse subtotal. Por exemplo, selecionando o subtotal de horas feitas por um motorista em viagens deve revelar as horas que fez em cada viagem, com ordenação decrescente (tendo em conta o período de tempo escolhido no início). Outro exemplo é a seleção do subtotal de viagens feitas por um motorista, que deve mostrar quando é que cada viagem desse motorista começou e terminou, da mais recente para a mais antiga (dentro do período escolhido inicialmente);

d) Deve ser possível selecionar qualquer viagem, táxi, ou motorista que apareça num relatório para obter mais detalhes.

User Story 13: Como gestor, quero poder **visualizar relatórios sobre clientes e faturação**. Os critérios de aceitação são os seguintes:

a) Deve ser possível visualizar o total de euros cobrados em viagens, tendo em conta um período de tempo, devendo por omissão ser o dia de hoje;

b) O total de euros deve poder ser selecionado, para mostrar os subtotais que explicam esse total. Assim, devem ser revelados quantos euros foram pagos por cada cliente, com ordenação decrescente (tendo em conta o período escolhido inicialmente);

c) Qualquer um dos subtotais deve poder ser selecionado, para mostrar os detalhes que explicam esse subtotal. Assim, selecionando o subtotal de euros pagos por um dado cliente deve revelar quantos euros pagou por cada viagem, com ordenação decrescente;

d) Deve ser possível selecionar qualquer viagem ou cliente que apareça num relatório para obter mais detalhes.

6 Requisitos adicionais

As tarefas de implementação das *user stories* da aplicação *web* devem ter em conta os requisitos não funcionais indicados nesta secção.

Responsive web design. A aplicação *web* deve adaptar as funcionalidades em função do tipo de dispositivo do utilizador, nomeadamente ecrãs de diferentes tamanhos e formas de interação (ex. com um rato e com um ecrã tátil).

Instalação da aplicação na infraestrutura do cliente. No último *sprint* do projeto, após estarem implementadas e testadas as *user stories* nos computadores de desenvolvimento do grupo, solicita-se que seja feita a instalação da aplicação no servidor appserver.alunos.di.fc.ul.pt, existindo mais instruções no Apêndice. Desta forma, o grupo pode passar pela experiência de instalar uma aplicação noutra infraestrutura computacional.

7 Recomendações

Seguem-se algumas recomendações sobre o trabalho a realizar.

Controlo de versões. Durante o desenvolvimento, deve ser utilizada uma plataforma Git, como a existente em git.alunos.di.fc.ul.pt. O uso de ferramentas de controlo de versões facilita o trabalho de uma equipa de várias pessoas e torna mais simples a passagem do código desenvolvido nos computadores de desenvolvimento do grupo para o servidor appserver.alunos.di.fc.ul.pt.

Design system. O grupo deve reutilizar um *design system* já existente, para desenvolver mais rapidamente uma interface apelativa, coerente, e familiar ao utilizador. Numa aula teórica, disponível no Moodle, o tema de *design systems* foi abordado, tendo sido mostrados vários exemplos, acompanhados de hiperligações para descarregar recursos e obter mais informações.

Desenho da API. A *Application Programming Interface* programada no *backend* em Express deve ser desenhada tendo em conta as boas práticas da indústria,[24] incluindo o formato dos dados dos pedidos e respostas, os nomes dos *endpoints*, os códigos de estado HTTP, entre outros aspetos.

8 Entregas

No final de cada *sprint*, cada grupo deve entregar um vídeo com a demonstração das *user stories* implementadas durante o *sprint*. Adicionalmente, cada aluno (individualmente) deve entregar um relatório de progresso listando as tarefas que concluiu durante o *sprint* e as tarefas que ficaram por concluir.

Os **vídeos**, com duração máxima de 2 minutos, devem ser entregues através de uma atividade no Moodle da unidade curricular. O nome do ficheiro deve

[24] https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design

seguir o formato PSI_Sx_TPyy_Gzz, onde x é o número do *sprint* (1, 2, ou 3), yy é o número da turma teórico-prática (ex. 21), e zz é o número do grupo (ex. 01).

Os **relatórios individuais**, com uma página apenas e em formato PDF,[25] devem ser entregues através de outra atividade no Moodle. O nome do ficheiro deve ter o formato PSI_Sx_TPyy_Gzz_nnnnn.PDF, onde x é o número do *sprint*, yy é o número da turma teórico-prática, zz é o número do grupo, e nnnnn é o número de aluno.

O fim dos *sprints* é nos seguintes dias, sendo iguais para todos os grupos:

- *Sprint* 1: ~~16~~ 24 de abril (quinta-feira);
- *Sprint* 2: ~~9~~ 11 de maio (domingo);
- *Sprint* 3: 23 de maio (sexta-feira).

9 Avaliações

O projeto é avaliado com base no cumprimento dos critérios de aceitação das *user stories* dos *sprints*, tendo todos os *sprints* o mesmo peso, bem como na discussão final com o grupo. A nota no projeto é atribuída individualmente.

Cada *sprint* é avaliado em laboratório na aula teórico-prática da turma em que o grupo está inscrito, nas seguintes semanas:

- Sprint 1: de 28 de abril a 2 de maio;
- Sprint 2: de 12 a 16 de maio;
- Sprint 3: de 26 a 30 de maio e de 2 a 4 de junho.

É obrigatória a presença de todas as pessoas do grupo durante as avaliações.

A avaliação de cada *sprint* consiste na demonstração das funcionalidades implementadas, podendo ser realizada num computador portátil e devendo ser coerente com o vídeo entregue.

No último *sprint*, para além da demonstração, tem lugar a discussão final com o grupo, razão pela qual as avaliações acontecem também nas aulas teóricas, estendendo-se por duas semanas.

10 Apêndice

Este apêndice tem instruções sobre como usar o servidor appserver.alunos.di.fc.ul.pt, que simula a infraestrutura de um cliente.

O acesso ao servidor é feito através de ssh sendo o *username* PSIXXX, em que PSI está em maiúsculas e XXX é o número do grupo (ex. PSI001). A *password* inicial é igual ao *username* e deve ser alterada após o primeiro *login*.

[25] Existe um modelo de relatório individual no Moodle da unidade curricular.

```
> ssh PSI001@appserver.alunos.di.fc.ul.pt
```

O appserver.alunos.di.fc.ul.pt tem as ferramentas node, npm e mongo. Todos os módulos necessários para o funcionamento do projeto devem ser instalados recorrendo ao npm.

Cada grupo tem uma base de dados e um *username* no servidor MongoDB. O nome da base de dados, o *username*, e a respetiva *password* têm o formato PSIXXX, em que PSI está em maiúsculas e XXX é o número do grupo (ex. PSI001).

Para abrir a consola do MongoDB pode ser usado o comando seguinte, substituindo XXX pelo número do grupo:

```
> mongosh --username PSIXXX --password --authenticationDatabase PSIXXX \
        appserver.alunos.di.fc.ul.pt/PSIXXX
```

Ou, simplesmente:

```
> mongosh --username PSIXXX --password --authenticationDatabase PSIXXX
```

A *connection string* para acesso à base de dados MongoDB deve ser a seguinte, em que XXX deve ser substituído pelo número do grupo:

```
mongodb://PSIXXX:PSIXXX@localhost:27017/PSIXXX?retryWrites=true&authSource=PSIXXX
```

Cada grupo tem dois portos disponíveis para acesso por HTTP a servidores node, um para o *frontend* e outro para o *backend*. O primeiro porto está no intervalo 3001 a 3050 e o segundo porto no intervalo 3051 a 3100. Por exemplo, o grupo 3 deve usar os portos 3003 e 3053.

A forma de executar o servidor node que serve o *frontend* em Angular deve ser a seguinte, onde YYYY é o porto, específico de cada grupo:

```
> ng serve --port YYYY --host 0.0.0.0 --disableHostCheck true
```

Para o servidor Node.js que serve o *backend* não é necessário mudar a forma de execução.

Devido a limitações de espaço no appserver.alunos.di.fc.ul.pt, eventuais imagens e vídeos usados na aplicação *web* devem estar alojados externamente.

Algumas soluções para situações de erro típicas podem ser encontradas no guião de apoio ao projeto, apresentado nas aulas teórico-práticas e disponível no Moodle.

Bom trabalho no projeto!