Aplicações para a Internet

Engenharia Informática	2º Ano	2º Semestre	2021-22	Avaliação Periódica
Projeto	Prazo para divulgação resultados: 4 Julho 2022			
Data: 22 Abril 2022	**Data de Entrega: 20 Junho 2022**			

Projeto – CineMagic

OBJETIVO

O objetivo deste projeto é implementar uma aplicação Web baseada no servidor, utilizando a Framework Laravel, para a empresa CineMagic que organiza sessões de cinema nas suas salas e que irá utilizar a aplicação Web para vender bilhetes e controlar as entradas nas sessões de cinema.

CENÁRIO

Os utilizadores (incluindo anónimos) da aplicação do cinema "CineMagic" poderão aceder à informação sobre os filmes em exibição nas salas da "CineMagic" (nome, cartaz, sumário, *trailer*, etc.), bem como à informação sobre cada uma das sessões (filme, data, hora, sala), incluindo os lugares ocupados e disponíveis para cada sessão.

Os clientes do "CineMagic" poderão registar-se na aplicação e por essa via comprar bilhetes para as sessões de cinema. As compras de bilhetes são feitas através de um carrinho de compras e o pagamento é feito online (o processo de pagamento é simulado). Os clientes deverão também aceder ao histórico dos seus recibos e bilhetes.

Os funcionários da "CineMagic" serão os responsáveis pelo controlo de acesso às sessões de cinema, confirmando através da aplicação se os bilhetes apresentados à entrada são válidos. Se os bilhetes forem válidos, permite o acesso à sessão e utiliza a aplicação para invalidar os bilhetes.

Os administradores da empresa "CineMagic" irão utilizar a aplicação para configurar alguns parâmetros de negócio, gerir as salas e respetivos lugares, gerir os filmes em exibição e respetivas sessões. São também responsáveis pela gestão de utilizadores, ou seja, criar, alterar e apagar os gestores e os funcionários, e bloquear/desbloquear clientes. Por último, terão também acesso a informação estatística sobre o negócio da empresa (bilhetes vendidos, percentagens de ocupação globais, por filme, por mês, por dia da semana, etc.).

FUNCIONALIDADES

A aplicação Web deverá implementar um conjunto de grupos de funcionalidades, que serão descritos de seguida. A implementação das mesmas deve ter em conta a informação descrita no cenário, na base de dados (na descrição do enunciado e na estrutura da própria base de dados) e na própria descrição dos grupos de funcionalidades, bem como todos os padrões e as boas práticas da Framework Laravel. Para além do modelo de negócio, requisitos e restrições que se podem inferir a partir do enunciado e da estrutura da base de dados, os estudantes têm liberdade para decidir sobre a interface com o utilizador e usabilidade, bem como sobre aspetos do negócio não descritos.

1. AUTENTICAÇÃO, PERFIL E GESTÃO DE UTILIZADORES

A aplicação Web a desenvolver deverá suportar autenticação através de um login com as credenciais e-mail + senha (password) – o processo de login é o mesmo para qualquer tipo de utilizador. Depois de autenticado qualquer utilizador pode alterar a senha (password) e sair da aplicação (logout). Caso o utilizador não se lembre da senha deverá ter a possibilidade de fazer o "*reset*" da senha – a aplicação deverá enviar um mail com um link para efetuar essa operação ("*password reset*").

Os utilizadores anónimos (utilizadores não autenticados), para além do acesso ao login para autenticação podem registar-se como clientes. O registo dos clientes é feito pela aplicação sem intervenção dos administradores ou funcionários. Depois de concluído o processo de registo, a aplicação irá enviar um e-mail para verificação/confirmação da validade de e-mail.

Os clientes têm acesso ao seu perfil de utilizador, onde podem consultar e alterar os seus dados pessoais – nome, NIF (opcional), dados de pagamento por omissão (opcional) e foto/avatar (opcional). Os funcionários não têm acesso ao seu perfil de utilizador, pois apenas os administradores podem consultar e alterar a sua informação pessoal.

Os administradores são os responsáveis pela gestão de utilizadores dos funcionários e dos próprios administradores, o que significa que podem consultar, filtrar, criar, alterar, bloquear/desbloquear ou remover as contas dos funcionários e administradores (exceto bloquear ou remover a ele próprio), bem como aceder e alterar o perfil de utilizador de qualquer funcionário ou administrador. Em contrapartida, mesmo os administradores não podem aceder aos perfis de utilizadores dos clientes – apenas podem consultar e filtrar a lista de clientes e bloquear/desbloquear ou apagar (usando o *soft delete*) as contas dos clientes.

Nota: os e-mails deverão ser enviados através do serviço mailtrap.io.

2. FILMES EM EXIBIÇÃO

Todos os utilizadores da aplicação, incluindo utilizadores anónimos (qualquer utilizador é anónimo antes de fazer o login) deverão conseguir consultar os filmes em exibição no cinema "CineMagic". Considera-se que um filme está em exibição, se tiver pelo menos uma sessão para o dia de hoje ou

qualquer dia posterior. Para cada filme a aplicação deverá apresentar o título, o cartaz, o sumário, um trailer (se existir), bem como as sessões do filme (só as sessões do dia de hoje ou dias posteriores), em que para cada sessão deverá ser apresentada a sala, data, hora de início e algo que indique se a sessão está esgotada ou não.

A aplicação deverá também permitir pesquisar (filtrar) filmes por género e por uma *sub-string* do nome ou do sumário.

3. COMPRA DE BILHETES

As compras de bilhetes são feitas através de um carrinho de compras, em que os utilizadores poderão adicionar ou remover bilhetes para qualquer sessão que tenha começado até há 5 minutos atrás ou que comece em data e hora posterior (por exemplo, se uma sessão começar às 15h00 de um determinado dia, o utilizador pode adicionar bilhetes para essa sessão até às 15h05 desse mesmo dia, independentemente da hora em que o carrinho de compras é concluído). Apesar de qualquer utilizador, incluindo anónimos, poder utilizar o carrinho de compras sem restrições, a aplicação deverá garantir que a confirmação final da compra só é feita por clientes registados.

O pagamento da compra poderá ser feito com cartão visa (16 dígitos + código CVC com 3 dígitos), Paypal (email) ou MB Way (nº telemóvel). O processo de pagamento é simulado (será fornecido código para simular o pagamento) e deverá ser feito imediatamente antes da compra ser finalizada. Caso o processo de pagamento seja concluído com sucesso, a aplicação vai criar um recibo e gerar os bilhetes relativos à compra, e vai limpar o carrinho de compras. Se o pagamento não for concluído com sucesso, a compra não é efetuada (nada é registado na base de dados) e o carrinho deverá manter-se tal como está, devendo o utilizador ser avisado que o pagamento é inválido.

A qualquer altura o utilizador pode apagar ou adicionar bilhetes ao carrinho de compras, ou limpar o carrinho de compras por completo (sem finalizar a compra).

Cada compra irá incluir um ou mais bilhetes e a informação necessária para criar o recibo relativo à compra, nomeadamente: a data da compra, os dados do pagamento, o NIF (opcional) e o nome do cliente, o total sem IVA, o IVA e o total com IVA. Cada bilhete deverá incluir um ID (que será usado como referência no controlo de acessos às sessões de cinema), a sessão (filme, sala, data e hora), o lugar reservado (localização na sala), o preço e uma referência para o cliente (cada bilhete fica associado a um cliente).

Note que o preço dos bilhetes e o valor do IVA será configurável na própria aplicação.

4. ESCOLHA DE LUGARES

Tal como referido anteriormente, cada bilhete ficará associado a uma sessão (um filme numa determinada sala e num determinado horário) e a um lugar específico na sala (para essa sessão). A aplicação deverá garantir que cada lugar de uma sessão está associado a bilhete único, ou a zero bilhetes enquanto esse lugar estiver vazio.

Quando um utilizador adicionar um bilhete ao carrinho de compras, deverá escolher qual o lugar associado ao bilhete de forma o mais apelativa possível, de preferência com uma representação visual de todos os lugares da sala, e incluindo uma forma de identificar os lugares ocupados e livres da sessão escolhida.

5. HISTÓRICO, RECIBOS E BILHETES

Depois de concluído com sucesso o processo de compra de bilhetes (incluindo o pagamento), a aplicação deverá registar o recibo associado à compra e esse recibo deverá ser enviado automaticamente por email para o cliente.

O recibo deverá estar sempre disponível em formato HTML, ou seja, ficará associada a um endereço URL que será imutável ao longo do tempo. Para além disso, a aplicação deverá gerar uma cópia do recibo em PDF, garantindo que a mesma é armazenada na aplicação e que ficará disponível ao longo do tempo.

O recibo deverá incluir o nº de recibo (ID automático), a data da compra, os dados do pagamento (tipo de pagamento e referência do pagamento), o NIF (opcional) e o nome do cliente, o total sem IVA, o IVA, o total com IVA e a lista de bilhetes associados à compra, em que para cada bilhete deverá incluir o ID do bilhete, o filme, a sala, a data e horário da sessão e o lugar associado ao bilhete.

Finalizado o processo de compra, para além de gerar o recibo da compra a aplicação deverá também gerar todos os bilhetes associados à compra e enviá-los automaticamente por email para o cliente - na mesma mensagem de email em que é enviado o recibo.

Tal como acontece com os recibos, os bilhetes deverão estar sempre disponíveis em formato HTML, ou seja, cada bilhete deverá ficar associado a um endereço URL. Em complemento a esse formato, a aplicação deverá também gerar cópias dos bilhetes em PDF, mas estas, ao contrário do que acontece com os recibos, não deverão ficar armazenadas na aplicação – o bilhete em PDF será gerado dinamicamente sempre que utilizador descarregar o mesmo.

Cada bilhete (em HTML ou PDF) deverá incluir o ID do bilhete, o filme, a sala, a data e horário da sessão, o lugar associado ao bilhete e o nome e foto/avatar (se existir) do cliente que comprou o bilhete. O bilhete em HTML deverá também incluir a informação que indique se o bilhete já foi usado ou não. Em contrapartida, o bilhete em PDF deverá incluir um "*QR code*" com o URL associado ao bilhete para efeitos de controlo de acesso à entrada da sessão.

Para além de receberem por email o recibo e os bilhetes quando concluída uma compra, os clientes deverão ter acesso ao histórico de todos os recibos relativos às compras que já efetuaram, devendo ser possível consultar (em HTML) ou descarregar (em PDF) os recibos e bilhetes associados aos recibos. Os clientes deverão também ter acesso aos bilhetes válidos, em que se considera um bilhete como válido quando o mesmo ainda não foi usado (estado = "não usado") e é relativo a uma sessão que decorre hoje ou irá decorrer no futuro.

6. CONTROLO DE ACESSO À SESSÃO

Os funcionários da "CineMagic" serão os responsáveis pelo controlo de acesso às sessões de cinema, confirmando através da aplicação se os bilhetes apresentados à entrada são válidos. Para tal, deverão ter acesso a uma página em que definem qual a sessão (um determinado filme, numa determinada sala, num determinado dia e horário) que estão a controlar. Quando um cliente apresenta um bilhete o funcionário introduz o ID (ou o URL) do bilhete na aplicação. Se o bilhete for inválido (porque não existe, porque é de uma sessão diferente ou porque já foi usado) a aplicação avisa o funcionário e este proíbe o acesso da pessoa à sessão. Se o bilhete for considerado válido para a sessão em questão, a aplicação deverá mostrar os detalhes do bilhete (incluindo o nome e foto do cliente) e informa o funcionário que o bilhete é válido – o funcionário pode ainda verificar a identidade da pessoa comparando com a informação (nome e foto) do cliente associado ao bilhete (o funcionário tem a liberdade de proibir o acesso a pessoas cujo bilhete está associado a outro cliente a não ser que seja comprovadamente um familiar do mesmo). Quando o funcionário decide validar a entrada com o referido bilhete, deverá clicar num botão da aplicação e a partir desse momento o bilhete passa a ficar inválido (estado passa de "não usado" para "usado").

Nota: numa aplicação real, a leitura do bilhete seria feita por um leitor de "*QR code*" (um leitor especializado ou por um telemóvel) que iria dar acesso a um URL relativo ao controlo de acessos do bilhete específico. Neste caso, podemos substituir a leitura do "*QR code*" pela introdução direta do URL que estaria associado ao "*QR code*" (em alternativa ao ID do bilhete).

7. ADMINISTRAÇÃO DO NEGÓCIO

Os administradores, para além da gestão dos utilizadores (*grupo de funcionalidades "1- Autenticação, perfil e gestão de utilizadores"*) são os responsáveis pela administração do negócio através da gestão das salas e lugares, filmes e sessões e através da definição de parâmetros de configuração como o preço atual dos bilhetes (sem IVA) e a percentagem de IVA a aplicar aos bilhetes vendidos.

Na gestão de salas deve ser possível criar, alterar ou apagar (através de *soft deletes*) as salas de cinema que a empresa dispõe, e para cada sala deverá ser possível definir os lugares que a sala tem. Valorizam-se soluções em que a definição dos lugares seja o mais simples possível – através de algum processo que automatize a criação dos lugares (em vez de criar lugares manualmente).

A gestão de filmes e sessões implica a possibilidade de criar, alterar e apagar filmes (só deverá ser possível apagar filmes que por alguma razão ainda não têm nenhuma sessão associada). Para cada filme deverá ser possível definir o título, género, sumário, cartaz (através de *upload* de um ficheiro JPG ou PNG), trailer (URL externo à aplicação com o endereço do vídeo) e um conjunto de sessões, em que cada sessão deverá ficar associada ao filme, a uma sala, a uma data e a um horário (hora de início de exibição do filme). As sessões só podem ser alteradas ou removidas se ainda não tiverem nenhum bilhete vendido para a sessão em questão.

Valorizam-se soluções em que a definição das sessões seja o mais simples possível, por exemplo, permitindo criar várias sessões em simultâneo para um determinado filme e sala, e para um conjunto de datas e horários que se repetem todos os dias (em vez de criar as sessões manualmente).

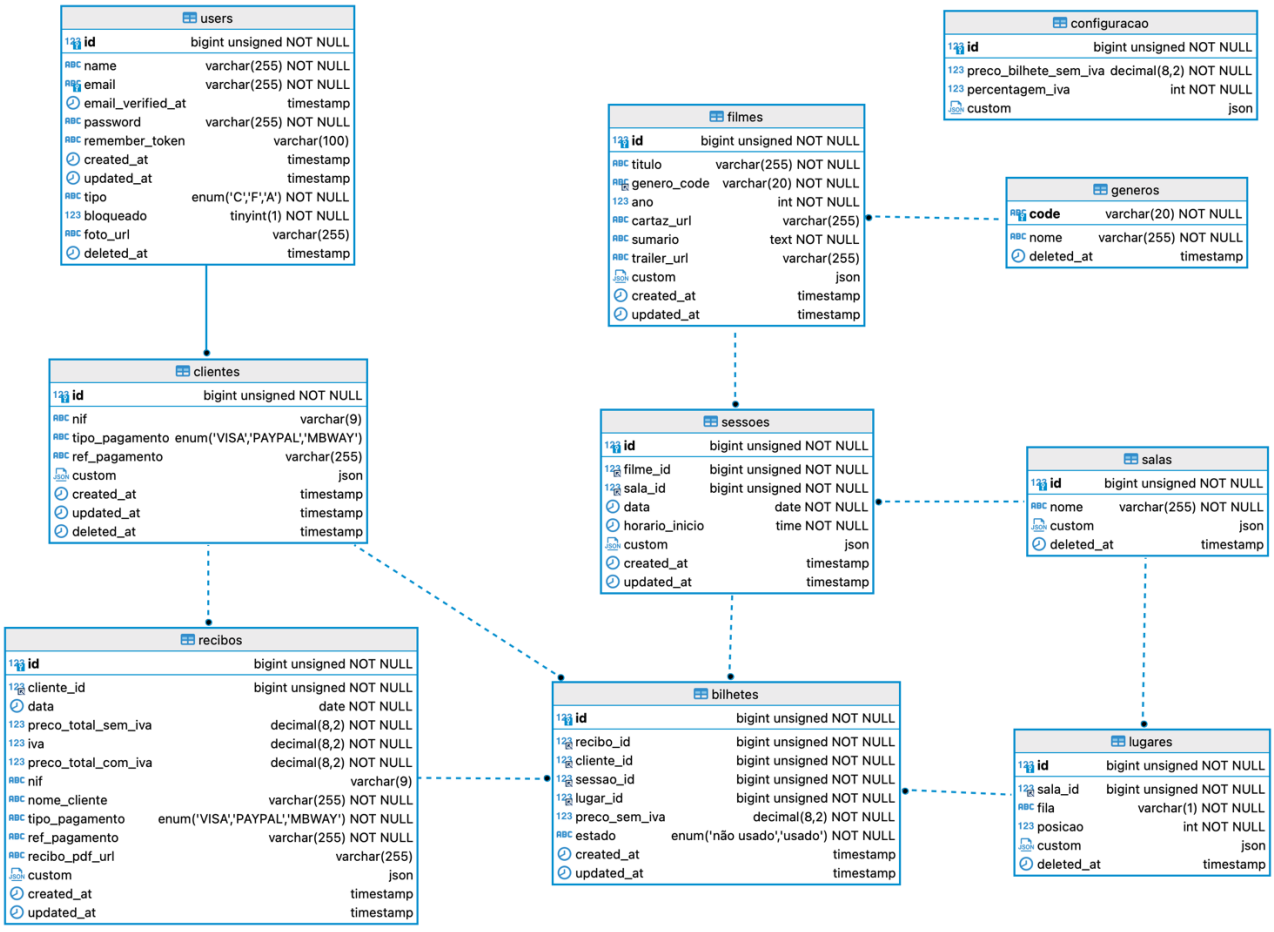
Nota: Os lugares livres de cada sessão são definidos indiretamente pelos lugares da sala que ainda não têm bilhetes vendidos para a sessão em concreto.

8. ESTATÍSTICAS

Os administradores deverão conseguir visualizar informação estatística relativa ao negócio do cinema "CineMagic". A decisão sobre a forma e o tipo de informação apresentada é da responsabilidade dos estudantes. Por exemplo, a informação pode ser apresentada em tabelas, gráficos ou de outras formas; pode ser exportada em ficheiros Excel, CSV ou outros formatos; pode incluir totais, médias, máximos, mínimos de vendas de bilhetes por valor ou quantidade, por mês, por ano, organizadas por filmes ou categorias de filmes, por cliente; pode incluir percentagens de ocupação das salas por filme, mês, dia da semana, globais, etc.

BASE DE DADOS

A estrutura da base de dados é fornecida (através de migrações) e é a seguinte:



A estrutura da base de dados não pode ser alterada pelos estudantes, exceto se explicitamente indicado pelo docente. Também serão fornecidos dados de exemplo através de "*database seeders*".

TABELAS E COLUNAS

De seguida são descritas as tabelas e as colunas da base de dados. A estrutura das tabelas e colunas (por exemplo, as chaves estrangeiras, os tipos de dados das colunas, se as colunas são ou não obrigatórias) e a descrição aqui apresentada, são relevantes para o funcionamento da aplicação e validação dos dados introduzidos pelos utilizadores.

USERS

Tabela com os utilizadores registados da aplicação - clientes, funcionários e administradores.

Esta tabela é uma superclasse da tabela clientes, o que significa que alguns utilizadores são clientes, e nesse caso os dados do cliente estão distribuídos pelas 2 tabelas: users e clientes. A tabela users terá todos os dados comuns a todos os users e a tabela clientes terá todos os dados específicos (exclusivos) dos clientes.

- "**id**" – chave primária – nº inteiro automático.
- "**name**" – nome do utilizador.
- "**email**" – e-mail do utilizador. O e-mail deverá ser único e serve também como credencial de autorização (em conjunto com a password).
- "**password**" – *hash* da password do utilizador.
- "**tipo**" – tipo de utilizador. C (cliente); F (funcionário) e A (administrador).
- "**bloqueado**" – booleano (1 ou 0) que indica se o utilizador está ou não bloqueado. Um utilizador bloqueado não deverá conseguir entrar na aplicação (autorização deverá falhar)
- "**foto_url**" (opcional) – nome ("relativo") do ficheiro com a foto (ou avatar) do utilizador.

Os próximos campos são geridos automaticamente pelo Laravel e normalmente não são apresentados na aplicação:

- ***"email_verified_at"*** *(opcional) – data em que o e-mail do utilizador foi verificado – esta coluna é gerida pelo sistema de autenticação do Laravel.*
- ***"remember_token"*** *(opcional) – para implementação da funcionalidade "remember me" – esta coluna é gerida pelo Laravel.*
- ***"created_at"*** *(opcional) – data e hora em que o registo foi criado – esta coluna é gerida pelo Laravel.*
- ***"updated_at"*** *(opcional) – data e hora em que o registo foi alterado pela última vez – esta coluna é gerida pelo Laravel.*
- ***"deleted_at"*** *(opcional) – data e hora em que o registo foi apagado (com soft delete) – esta coluna é gerida pelo Laravel.*

CLIENTES

Tabela com os clientes registados do cinema.

Esta tabela é uma subclasse da tabela users, o que significa que os clientes são também users. Os dados do cliente estão distribuídos pelas 2 tabelas: users e clientes. A tabela users terá os dados que são comuns a todos os users (por exemplo, o nome do cliente está na tabela users) e a tabela clientes terá os dados específicos (exclusivos) dos clientes.

- "**id**" – chave primária – valor inteiro igual ao "id" do user. O "id" da tabela clientes não é preenchido automaticamente, terá sempre o mesmo valor que o "id" do user (o cliente é um user). Neste caso, o "id" do cliente é também a chave estrangeira responsável pelo relacionamento da tabela clientes com a tabela users.

  Nota: quando se insere um cliente é sempre necessário inserir primeiro um registo na tabela "users" e de seguida o registo equivalente na tabela "clientes". O contrário deverá acontecer se for necessário apagar um cliente: primeiro apaga-se o registo da tabela "clientes" e de seguida o registo equivalente na tabela "users".
- "**nif**" (opcional) – NIF (Número de Identificação Fiscal) do cliente. O nif (quando preenchido) deverá ter sempre 9 dígitos – por exemplo: 148928093. O nif da tabela clientes será usado para pré-preencher o nif dos recibos do cliente – corresponde ao valor por omissão do nif.
- "**tipo_pagamento**" (opcional) – Tipo de pagamento usado por omissão nos recibos do cliente. Os valores aceites são: "VISA" (cartão de crédito Visa); "PAYPAL" (Paypal); e "MBWAY" (MB Way)
- "**ref_pagamento**" (opcional) – Referência de pagamento usado por omissão nos recibos do cliente. Se o tipo de pagamento for "VISA", então o valor da referência de pagamento corresponde ao número do cartão de crédito e deverá ter 16 dígitos. Se o tipo de pagamento for "PAYPAL" então o valor da referência de pagamento corresponde ao e-mail da conta do Paypal. Se o tipo de pagamento for "MBWAY" então o valor de referência de pagamento deverá ser um nº de telemóvel em Portugal (9 dígitos, sendo que o primeiro dígito será sempre 9). Se o tipo de pagamento não for preenchido (valor = null) então a referência de pagamento também não deverá ser preenchida (valor = null).
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

Os próximos campos são geridos automaticamente pelo Laravel e normalmente não são apresentados na aplicação:

- ***"created_at"*** *(opcional) – data e hora em que o registo foi criado – esta coluna é gerida pelo Laravel.*
- ***"updated_at"*** *(opcional) – data e hora em que o registo foi alterado pela última vez – esta coluna é gerida pelo Laravel.*
- ***"deleted_at"*** *(opcional) – data e hora em que o registo foi apagado (com soft delete) – esta coluna é gerida pelo Laravel.*

CONFIGURACAO

Tabela com os parâmetros de configuração da aplicação. Esta tabela deverá ter um e um só registo.

- "**id**" – chave primária – nº inteiro automático.

  Nota: como a tabela tem sempre uma linha e uma linha apenas, este campo é irrelevante. No entanto, o facto de se definir uma chave primária facilita a integração da tabela com os modelos do Laravel.
- "**preco_bilhete_sem_iva**" – preço atual de cada bilhete (sem IVA).
- "**percentagem_iva**" – valor (em percentagem) atual do IVA.
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

GENEROS

Tabela com os géneros de filmes.

- "**code**" – chave primária – código do género (exemplos: DRAMA; COMEDY; SCIFI; etc.)
- "**nome**" – nome do género.

O próximo campo é gerido automaticamente pelo Laravel e normalmente não é apresentado na aplicação:

- ***"deleted_at"*** *(opcional) – data e hora em que o registo foi apagado (com soft delete) – esta coluna é gerida pelo Laravel.*

FILMES

Tabela com os filmes. Considera-se que um filme está em exibição se estiver associado a uma sessão que decorre no dia de hoje ou em data posterior.

- "**id**" – chave primária – nº inteiro automático.
- "**titulo**" – Título do filme.
- "**genero_code**" – Género do filme. Chave estrangeira (relacionada com a tabela generos) – identifica o género do filme.
- "**cartaz_url**" (opcional) – nome ("relativo") do ficheiro com a imagem (JPG ou PNG) do cartaz do filme. A imagem do cartaz deve ser armazenada na pasta "*storage*" da aplicação.
- "**sumario**" – Texto com o sumário (sinopse) do filme.
- "**trailer_url**" (opcional) – URL absoluto com o endereço do vídeo com o trailer deste filme. Ao contrário do que acontece com o cartaz, o trailer será sempre externo à aplicação (a aplicação não guarda os vídeos com os trailers)
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

Os próximos campos são geridos automaticamente pelo Laravel e normalmente não são apresentados na aplicação:

- *"**created_at**" (opcional) – data e hora em que o registo foi criado – esta coluna é gerida pelo Laravel.*
- *"**updated_at**" (opcional) – data e hora em que o registo foi alterado pela última vez – esta coluna é gerida pelo Laravel.*

SALAS

Tabela com as salas de cinema que a empresa "CineMagic" disponibiliza.

- "**id**" – chave primária – nº inteiro automático.
- "**nome**" – nome da sala.
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

O próximo campo é gerido automaticamente pelo Laravel e normalmente não é apresentado na aplicação:

- *"**deleted_at**" (opcional) – data e hora em que o registo foi apagado (com soft delete) – esta coluna é gerida pelo Laravel.*

LUGARES

Tabela com os lugares de uma sala. Cada lugar (cadeira) de uma sala corresponde a um registo desta tabela. Os bilhetes vendidos são associados a um lugar e não deverá ser possível associar vários bilhetes para o mesmo lugar na mesma sessão. Considera-se que um lugar de uma sessão está vazio se não houver bilhetes associados a esse lugar na referida sessão.

- "**id**" – chave primária – nº inteiro automático.
- "**sala_id**" – chave estrangeira (relacionada com a tabela salas) – identifica a sala a que pertence este lugar.
- "**fila**" – fila do lugar. Normalmente a fila é representada por uma letra (ex: D)
- "**posição**" – posição do lugar dentro da fila. A posição é definida por um número (ex: 3).
    - Se juntarmos a fila e posição, podemos ter uma representação do lugar através de uma letra e um número, por exemplo: "D3"
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

O próximo campo é gerido automaticamente pelo Laravel e normalmente não é apresentado na aplicação:

- *"**deleted_at**" (opcional) – data e hora em que o registo foi apagado (com soft delete) – esta coluna é gerida pelo Laravel.*

SESSOES

Tabela com as sessões. Cada sessão é relativa a um filme, que será exibido numa determinada sala, num determinado dia e num determinado horário.

- "**id**" – chave primária – nº inteiro automático.
- "**filme_id**" – chave estrangeira (relacionada com a tabela filmes) – identifica o filme associado à sessão.
- "**sala_id**" – chave estrangeira (relacionada com a tabela salas) – identifica a sala associada à sessão (onde o filme será exibido).
- "**data**" – dia em que a sessão decorre (em que o filme é exibido).
- "**horário_inicio**" – hora (ex: 20:30) em que a sessão tem início (começa a exibição do filme).
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

Os próximos campos são geridos automaticamente pelo Laravel e normalmente não são apresentados na aplicação:

- ***"created_at"*** *(opcional) – data e hora em que o registo foi criado – esta coluna é gerida pelo Laravel.*
- ***"updated_at"*** *(opcional) – data e hora em que o registo foi alterado pela última vez – esta coluna é gerida pelo Laravel.*

RECIBOS

Tabela com os recibos. Cada compra concluída e paga (processo de pagamento finalizado com sucesso) tem obrigatoriamente um recibo. Tendo em consideração esse facto, a tabela dos recibos poderá também ser usada para obter a informação sobre todas as compras registadas na aplicação.

Nota: enquanto o carrinho de compras estiver a ser construído, não deverá ser registado nenhum recibo. Só depois do cliente confirmar o carrinho de compras e da operação de pagamento ser executada e considerada como válida, é que a aplicação irá gerar os bilhetes e registar o recibo – o carrinho de compras é mantido na sessão do servidor e não na base de dados.

- "**id**" – chave primária – nº inteiro automático.
- "**cliente_id**" – chave estrangeira (relacionada com a tabela clientes) – identifica o cliente da compra associada ao recibo.
- "**data**" – data da compra – é apenas necessário o dia (a hora não interessa).
- "**preco_total_sem_iva**" – preço total da compra associada ao recibo sem IVA (multiplicação entre o total de bilhetes e o preço de cada bilhete – definido na tabela "configuracao", campo "preco_bilhete_sem_iva").
- "**iva**" – valor total do IVA (calcular esse valor tendo em conta o "preco_total_sem_iva" do recibo e o valor do IVA que está definido na tabela "configuracao", campo "percentagem_iva").

- "**preco_total_com_iva**" – preço total da compra associada ao recibo já com IVA ("preco_total_sem_iva" + "iva").
- "**nif**"– (opcional) NIF (Número de Identificação Fiscal) do cliente. O NIF (quando preenchido) deverá ter sempre 9 dígitos – por exemplo: 148928093. Se o cliente tiver um valor para o NIF, então esse valor deverá ser usado por omissão quando esse cliente estiver a preencher/confirmar a compra. No entanto, o cliente deverá ter sempre a liberdade de escolher outro NIF.
- "**nome_cliente**"– Nome do cliente. Quando o cliente estiver a preencher/confirmar a compra, este campo deverá ter por omissão o nome do cliente, mas o cliente deverá ter sempre a liberdade de escolher outro nome.
- "**tipo_pagamento**" – Tipo de pagamento usado na compra associada ao recibo. Os valores aceites são: "VISA" (cartão de crédito Visa); "PAYPAL" (Paypal); e "MBWAY" (MB Way). Se o cliente tiver um valor para o tipo_pagamento, então esse valor deverá ser usado por omissão quando esse cliente estiver a preencher/confirmar a compra. No entanto, o cliente deverá ter sempre a liberdade de escolher outro tipo_pagamento.
- "**ref_pagamento**" – Referência de pagamento usado na compra associada ao recibo. Se o tipo de pagamento for "VISA", então o valor da referência de pagamento corresponde ao número do cartão de crédito e deverá ter 16 dígitos. Se o tipo de pagamento for "PAYPAL" então o valor da referência de pagamento corresponde ao e-mail da conta do Paypal. Se o tipo de pagamento for "MBWAY" então o valor de referência de pagamento deverá ser um nº de telemóvel em Portugal (9 dígitos, sendo que o primeiro dígito será sempre 9). Se o cliente tiver um valor para a ref_pagamento, então esse valor deverá ser usado por omissão quando esse cliente estiver a preencher/confirmar a compra. No entanto, o cliente deverá ter sempre a liberdade de escolher outra ref_pagamento.
- "**recibo_pdf_url**" (opcional) – nome do ficheiro com o PDF relativo ao recibo (se a geração do recibo em PDF estiver implementada). Se valor é null, significa que o PDF não foi gerado. Depois de gerado o PDF, este deverá ser armazenado na pasta *storage* da aplicação para futura consulta – o PDF só deve ser gerado uma vez (considera-se que o documento é imutável).
- "**custom**" – objeto JSON que os estudantes poderão usar para armazenar o que quiserem.

Os próximos campos são geridos automaticamente pelo Laravel e normalmente não são apresentados na aplicação:

- ***"created_at"*** *(opcional) – data e hora em que o registo foi criado – esta coluna é gerida pelo Laravel.*
- ***"updated_at"*** *(opcional) – data e hora em que o registo foi alterado pela última vez – esta coluna é gerida pelo Laravel.*

BILHETES

Tabela com os bilhetes. Cada compra concluída e paga (processo de pagamento finalizado com sucesso) irá gerar um conjunto de bilhetes, em que cada bilhete corresponde a um determinado lugar numa determinada sessão.

Considera-se um bilhete válido quando o mesmo ainda não foi usado (estado = "não usado") e é relativo a uma sessão que decorre hoje ou irá decorrer no futuro.

- "**id**" – chave primária – nº inteiro automático.
- "**recibo_id**" – chave estrangeira (relacionada com a tabela recibos) – identifica o recibo associado ao bilhete. Com esta relação é possível identificar todos os bilhetes da compra associada a um determinado recibo.
- "**cliente_id**" – chave estrangeira (relacionada com a tabela clientes) – identifica o cliente que comprou o bilhete.
- "**sessao_id**" – chave estrangeira (relacionada com a tabela sessoes) – identifica a sessão associada ao bilhete.
- "**lugar_id**" – chave estrangeira (relacionada com a tabela lugares) – identifica o lugar associado ao blihete.
- "**preco_sem_iva**" – preço de venda (sem IVA) do bilhete. Uma vez que o preço dos bilhetes pode variar ao longo do tempo, este campo permite saber o preço concreto do bilhete quando o mesmo foi vendido (que poderá não ser o mesmo que o preço atual).
- "**estado**" – indica se o bilhete já foi usado para entrar na sessão (funcionário confirmou o bilhete, deixou a pessoa entrar na sessão e invalidou o bilhete). Valor pode ser "não usado" ou "usado". Note que um bilhete que não foi usado, mas é relativo a uma sessão que já decorreu, é na prática um bilhete inválido pois não poderá ser usado – cada bilhete só pode ser usado na sessão a que está associado.

Os próximos campos são geridos automaticamente pelo Laravel e normalmente não são apresentados na aplicação:

- ***"created_at"** (opcional) – data e hora em que o registo foi criado – esta coluna é gerida pelo Laravel.*
- ***"updated_at"** (opcional) – data e hora em que o registo foi alterado pela última vez – esta coluna é gerida pelo Laravel.*

SIMULAÇÃO DE PAGAMENTO

O processo de pagamento é simulado através da classe Payment (fornecida nos recursos do projeto) que deverá ser colocada na pasta "app/services". Esta classe inclui 3 métodos estáticos que devolvem *true* se o pagamento for considerado válido e *false* se o pagamento for considerado inválido. Os métodos da classe são os seguintes:

- **payWithVisa** ($card_number, $cvc_code) – Simula o pagamento com cartão visa;
- **payWithPaypal** ($emailAddress) – Simula o pagamento com Paypal;
- **payWithMBway** ($phone_number) – Simula o pagamento com MB way;

ENTREGA

A entrega do projeto deverá incluir 2 ficheiros relativos ao grupo acrescido de 1 ficheiro por cada elemento do grupo. Por exemplo, se o grupo tiver 3 elementos deverão ser entregues 5 ficheiros (2 relativos ao grupo e 3 individuais). Os ficheiros a entregar são os seguintes:

- 1 Ficheiro Excel com o **relatório do grupo** – o relatório do grupo vai incluir a identificação dos elementos do grupo e informação sobre a implementação do projeto. Este ficheiro deverá ser entregue apenas por um dos elementos do grupo. O modelo para o relatório será fornecido pelo professor;
- 1 Ficheiro Zip com todo o código do **projeto** – este ficheiro zip inclui uma cópia de todos os ficheiros e pastas do projeto, **exceto** as pastas:
    - "**.git**"
    - "**vendor**"
    - "**node_modules**"
    - "**database**"
    - "**storage/app/public/fotos**"
    - "**storage/app/public/cartazes**"

  Este ficheiro deverá ser entregue apenas por um dos elementos do grupo
- Vários ficheiros Excel com **relatórios individuais** – o relatório individual irá incluir a autoavaliação e avaliação entre pares. Todos os estudantes terão que entregar um relatório individual. O modelo para o relatório será fornecido pelo professor.

AVALIAÇÃO

Os critérios de avaliação do projeto são definidos pelos grupos de funcionalidades identificados anteriormente. Os pesos dos grupos de funcionalidades são os seguintes:

Nº	Peso	Grupo de funcionalidades
1	15%	Autenticação, perfil e gestão de utilizadores
2	15%	Filmes em exibição
3	15%	Compra de bilhetes
4	10%	Escolha de lugares
5	10%	Histórico, recibos e bilhetes
6	10%	Controlo de acesso à sessão
7	15%	Administração do negócio
8	10%	Estatísticas

A avaliação de cada um dos grupos de funcionalidades depende da quantidade e qualidade de funcionalidades implementadas, da integração das funcionalidades na aplicação e da usabilidade e correção (funcionamento correto) das funcionalidades. Serão também tidos em conta o layout e aspeto visual (aspeto consistente e com layout bem estruturado) e alguns requisitos não funcionais e transversais a toda a aplicação, tais como (entre outros):

- A implementação da aplicação deverá seguir o padrão arquitetural MVC (*Model-View-Controller*) e a estrutura, padrões de desenho e as boas práticas definidas na Framework Laravel;
- Os métodos (GET, POST, PUT, PATCH, DELETE) e URL das rotas deverão seguir as boas práticas das aplicações Web;
- A aplicação deverá usar as funcionalidades e componentes do Laravel adequados, para resolver os vários desafios da aplicação. Por exemplo, usar sempre que possível o Eloquent (em vez da classe DB) para aceder à base de dados; usar "Form Request" para implementar validações no servidor, usar o sistema de autenticação fornecido pelo Laravel (em vez de implementar um sistema próprio), usar o método de hash fornecido pelo Laravel para guardar as passwords, usar Laravel *policies* para autorização, etc;
- A aplicação deverá seguir o princípio DRY (*Dont't Repeat Yourself*), ou seja, deve sempre que possível evitar a repetição de código, sem, no entanto, quebrar as boas práticas do Laravel. Por exemplo, utilizar vistas parciais sempre que houver secções de HTML que se repitam em várias vistas ou acrescentar métodos aos modelos que devolvem determinado conjunto de dados que serão utilizados em vários controladores;
- A aplicação deverá ter o melhor desempenho possível. O cumprimento deste requisito depende em grande parte da forma como é feita a interação com a base de dados. No que concerne ao desempenho do acesso à base de dados, dois grandes princípios deverão nortear a implementação da aplicação: reduzir ao máximo o nº de comandos que são executados sobre a base de dados e garantir que cada consulta à base de dados traz apenas a informação necessária (só as colunas e as linhas estritamente necessárias).

A aplicação deverá também ter em consideração outras características que melhorem o desempenho geral da aplicação, tais como: utilização de *queues* para operações mais pesadas; utilização adequada de sistemas de cache (pesquisar sobre cache no Laravel); garantir que os recursos devolvidos pelo servidor têm o menor tamanho possível (reduzir o tamanho das respostas HTTP), etc;

- A aplicação deverá garantir o máximo de segurança e privacidade aos seus utilizadores. A aplicação tem que garantir que todos as operações e recursos que disponibiliza, são apenas acessíveis a quem está autorizado a fazê-lo. A informação privada dos utilizadores deverá estar sempre protegida de acessos indevidos.

  A implementação dos mecanismos que garantem a segurança, privacidade e autorização, deverá ser feita de acordo com as boas práticas do Laravel e utilizando as funcionalidades e componentes do Laravel adequados.

  > *Nota: uma das principais características que tornam as aplicações web mais seguras e com melhor proteção da privacidade, tem a ver com a utilização de um protocolo seguro para comunicação entre o cliente e o servidor – utilização do HTTPS em vez do HTTP. Como neste projeto estamos a tratar apenas da camada aplicacional do sistema e não da camada da comunicação, NÃO é necessário a utilização do HTTPS.*