Relatório do projeto da fase 1 de Construção de Sistemas de Software

Grupo 20

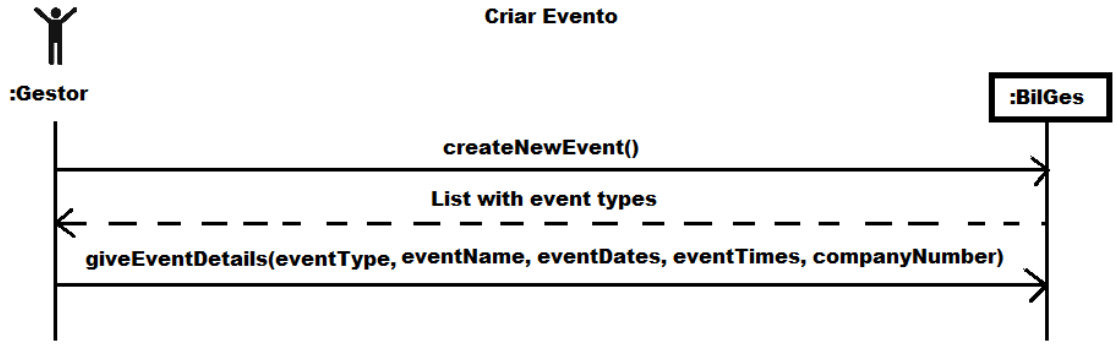
André Costa 52788

David Rodrigues 53307

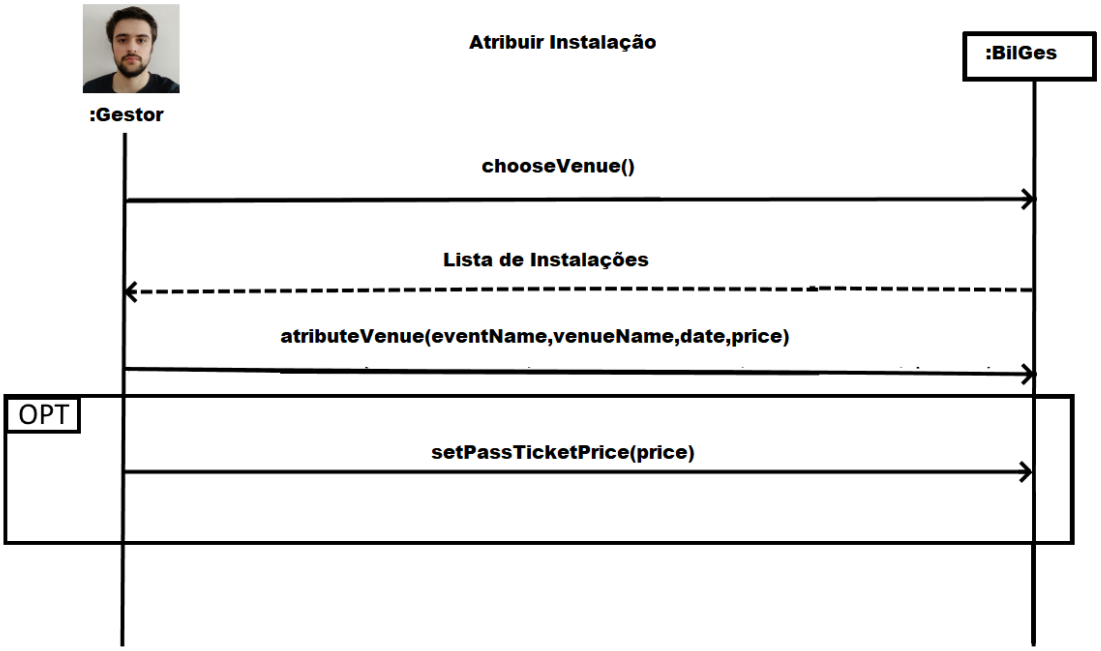
Martim Silva 51304

System Sequence Diagrams

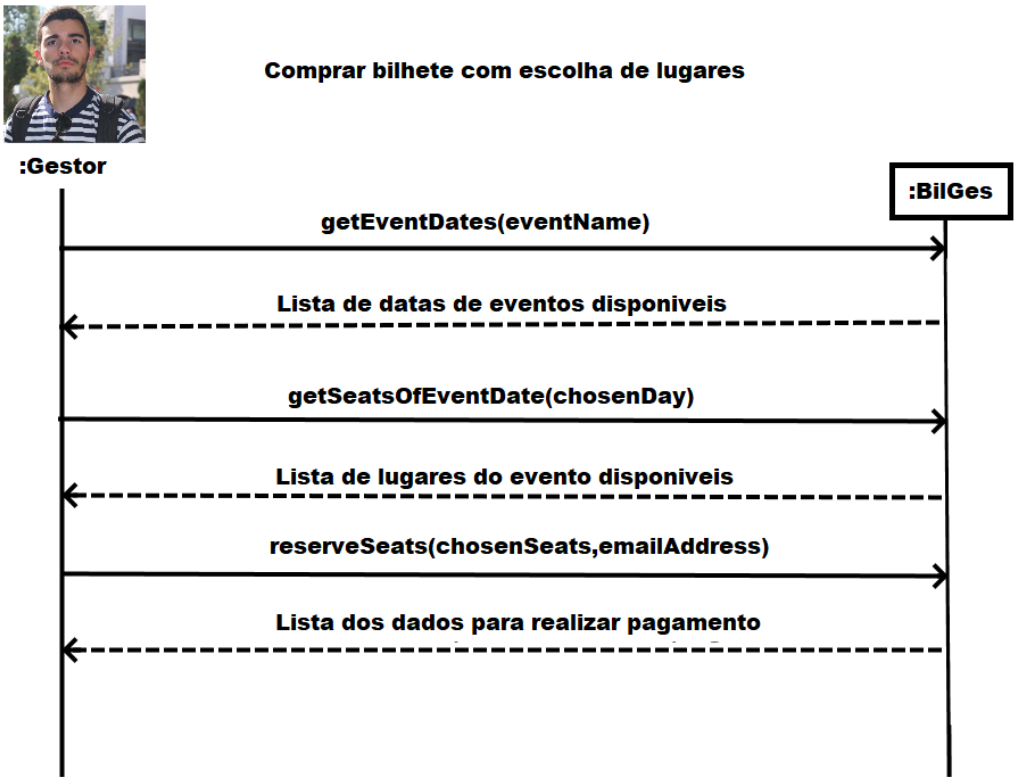
Caso de uso de criar evento



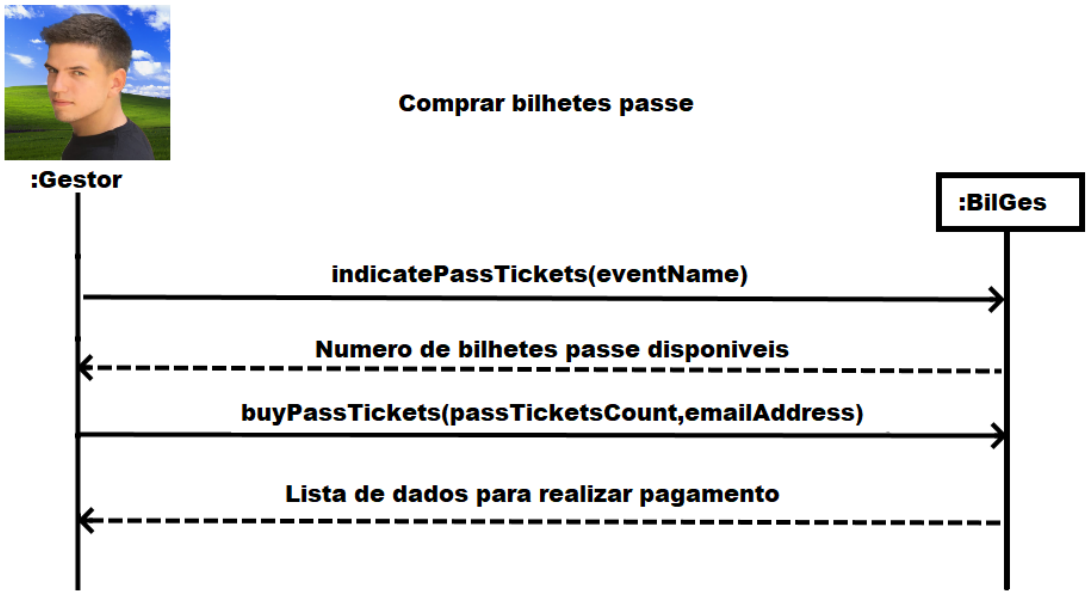
Caso de uso de atribuir instalação



Caso de uso de comprar bilhetes com escolha de lugares

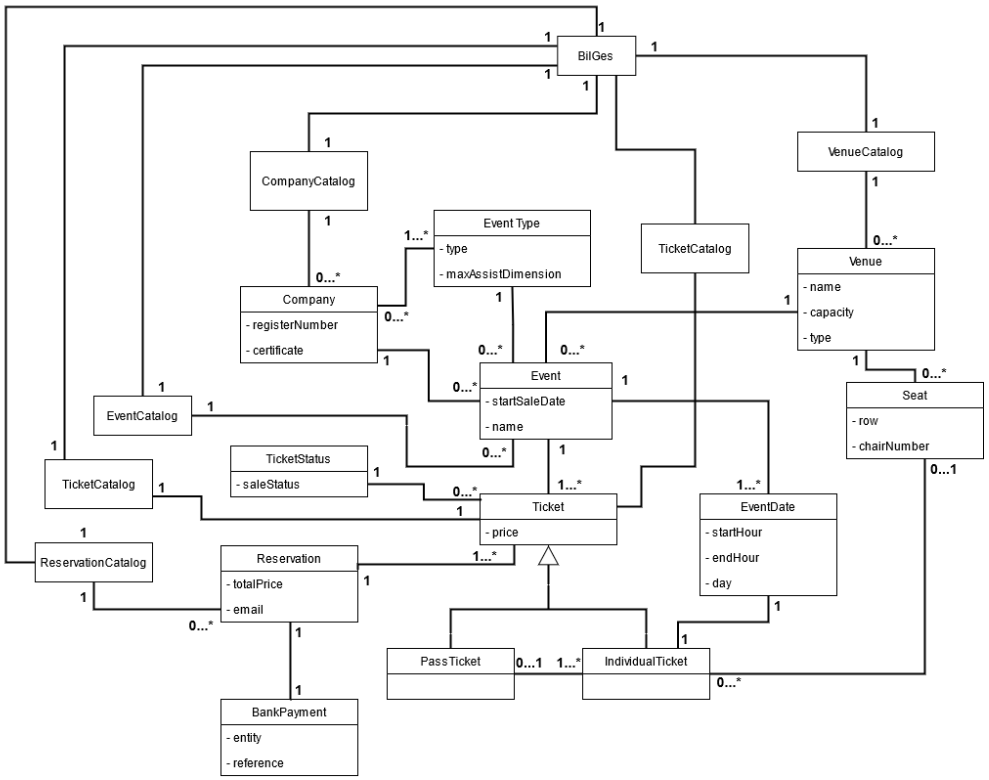


Caso de uso de comprar bilhetes passe



Modelo de Domínio

Conceitos
-Evento
-TipoEvento
-Instalacao
-Bilhete
-BilhetePasse
-BilheteIndividual
-LugarSentado
-DataEvento
-Empresa
-Gestor
-Utilizador
-Reserva
-PagamentoMB



Modelo de Classes

Como a fotografia é demasiado grande colocámo-la num ficheiro png à parte.

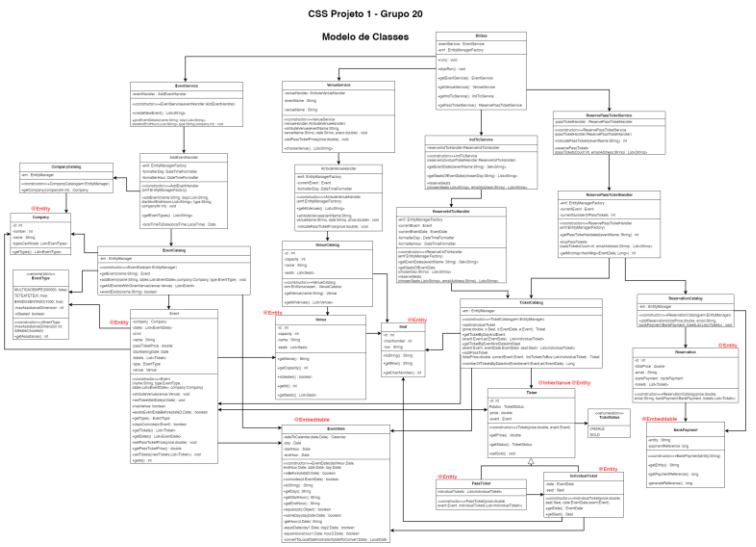
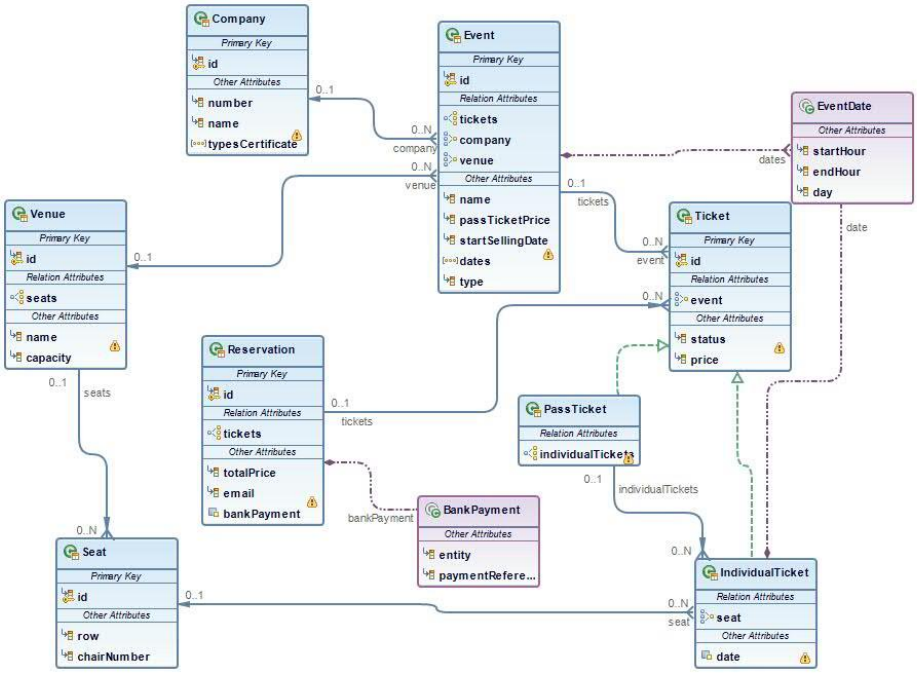


Diagrama da solução recorrendo ao JPA



Decisões importantes tomadas em termos de desenho da aplicação

Em relação à diferenciação entre lugares marcados e lugares não-marcados escolhemos ter apenas um objeto a identificar um lugar marcado (sentado) com os atributos que fazem sentido estar num lugar específico de uma instalação – fila e nº da cadeira.

Decidimos criar a classe **EventDate** para representar os atributos de um dia de um evento, isto é, um dia, uma hora de começo e uma hora de fim. Se o evento tem vários dias, terá várias **EventDates**. Um bilhete individual corresponde a um dia do evento logo tem uma **EventDate**.

Para o **primeiro caso de uso** temos que criar um evento, com uma dada empresa responsável e com um dado tipo de evento. Logo, vamos criar um novo evento com esse tipo, ficando o evento a conhecer a empresa produtora.

Para o **segundo caso de uso** atribuímos uma instalação, uma data de começo de venda dos bilhetes ao evento e um preço de bilhetes individuais, logo fazemos com que a instalação pedida conheça o evento e geramos um bilhete individual para cada lugar dessa instalação.

De notar que independentemente de ser uma instalação com lugares marcados ou não, geramos bilhetes, a diferença é que bilhetes para lugares sentados conhecem um lugar de uma dada instalação, mas bilhetes para lugares em pé apenas correspondem a um dado evento.

Em suma, cada bilhete corresponde a uma data, logo são gerados bilhetes para cada lugar da instalação, para todas as datas do evento. Existe também a possibilidade de o cliente permitir bilhetes passe para o evento, neste caso, apenas guardamos o seu preço no evento.

No futuro, se um cliente quiser comprar bilhetes passe, é computada a quantidade de bilhetes passe disponíveis através do número de bilhetes individuais existentes para cada dia do evento e é gerado o bilhete já como vendido.

Para o **terceiro caso de uso**, queremos comprar bilhetes individuais que correspondem a lugares. Depois de indicar o nome e a data do evento, o cliente escolhe um (ou mais) lugares de uma lista de lugares disponíveis e o sistema trata de atualizar o estado desse(s) bilhete(s) como vendido(s) e cria uma reserva com o preço total e toda a informação necessária para fazer o pagamento.

No **quarto caso de uso** permitimos a compra de bilhetes passe. Após o cliente indicar que pretende comprar bilhetes passe indicando o nome do evento, o sistema indica quantos bilhetes passe estão disponíveis e escolhe um bilhete individual para cada dia para cada bilhete passe pretendido.

Detalhes:

- Uma empresa tem uma lista de tipos de evento que está certificada a fazer.

- As horas de uma **EventDate** são do tipo "**Date**", logo terão um dia na base de dados, mas esse dia é ignorado.

- As implementações da classe abstrata **Ticket** são persistidas numa Single Table, logo como um bilhete passe não tem **EventDates**, os atributos da mesma são nullable para permitir a persistência de bilhetes passe.

Temos também definida entre **Event** e **Ticket** uma relação bidirecional em que um evento conhece um conjunto de bilhetes e cada bilhete pertencente a esse evento conhece apenas esse evento e quem mapeia esta relação é o evento no contexto do JPA.