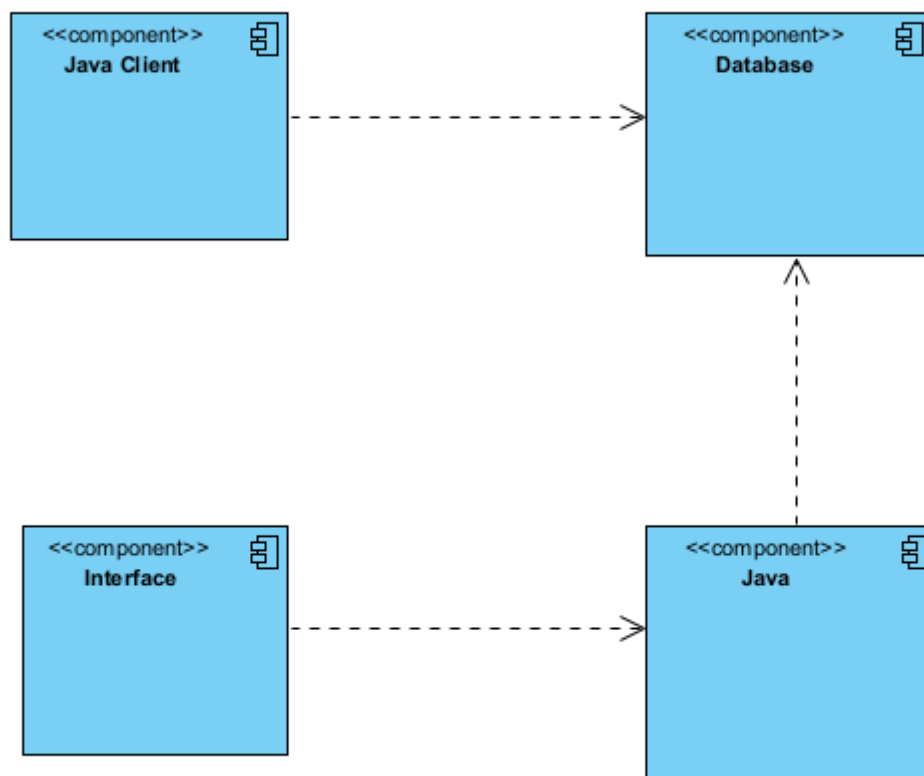

Relatório de Implementação

Sumário: xxx
Grupo: xxx (Nº xxx)
xxx (Nº xxx)
Data de preparação: 2004-09-22
Versão: v1.1
Circulação: xxx

Índice

1	Modelo de componentes da solução.....	3
2	Instalação	5
2.1	Modelo de instalação.....	5
2.2	Manual de instalação.....	5
3	Testes e validação	7
4	Anexos.....	10

1 Modelo de componentes da solução



O nosso programa é composto por 7 classes públicas a partir das quais são criados objetos a representar entidades com as quais o utilizador interage. Nomeadamente, estas classes são Flight (Voo), Ticket (Bilhete), Passenger (Passageiro), Class (Classe), Crew (Tripulação), Seat (Assento) e, por último, CheckIn, que representa a atividade de check-in.

Para além destas, existem ainda 9 classes responsáveis pela forma como o utilizador interage com a aplicação, através da interface gráfica desta. 2 destas classes são destinadas a ser utilizadas pelo passageiro, uma primeira como menu de inicialização e a segunda para comprar o bilhete para a viagem. Esta segunda é também o componente mais complexo da aplicação, utilizando objetos de tipo Flight, Class, Seat, Passenger, Ticket e CheckIn.

As outras 7 classes foram desenvolvidas para uso do administrador, com 1 delas a ser um menu de inicialização semelhante ao do passageiro e outras 3 criadas com a administração de recursos da companhia em mente: 1 para gestão de voos, outra para gestão da tripulação de cada voo e uma última para gerir classes. Elas trabalham com objetos do tipo Flight, Crew e Class, respetivamente.

O menu de inicialização do administrador tem 3 botões; ao contrário do menu do passageiro, que leva apenas para a página onde este pode realizar a compra de bilhetes, este menu permite ao administrador consultar voos, adicionar voos através da interface para

gestão de voos mencionada previamente e gerir classes através da interface para gestão de classes também mencionada acima.

Quanto à interface para gestão de tripulação, esta é acedida através da interface para gestão de voos, ao clicar num dado voo, o que permite gerir os tripulantes deste.

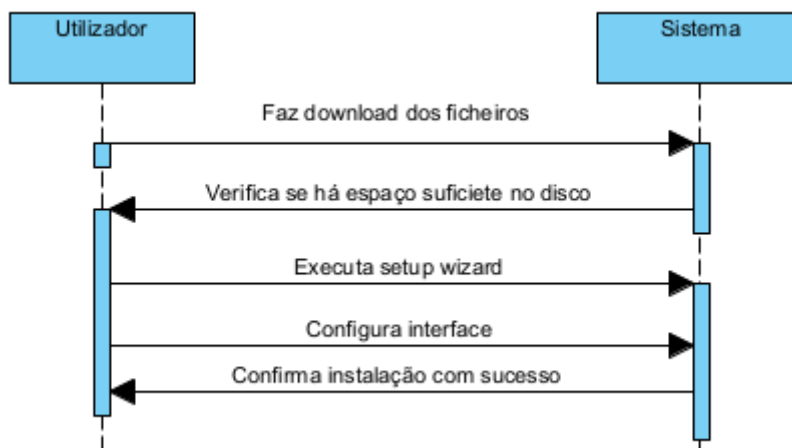
Em relação às 3 últimas classes, estas são classes destinadas a representar as 3 classes de voo disponíveis para ser escolhidas pelo passageiro, com diferentes opções de assento e serviços oferecidos a este durante o voo.

A informação utilizada pela aplicação é enviada e subsequentemente guardada numa base de dados, de forma a permitir uma maior segurança e fiabilidade no que toca à informação dos utilizadores. Ela guarda identificadores para todos os objetos, assim outros certos dados relevantes, como o preço de bilhetes, a origem e destino de cada voo e a hora na qual o check-in foi realizado pelo passageiro.

2 Instalação

2.1 Modelo de instalação

[diagrama de instalação]



[descrição dos nodos]

In

2.2 Manual de instalação

Este manual descreve os passos necessários para instalar o sistema Skybound, incluindo os requisitos de hardware e software, e os procedimentos para a configuração inicial.

Primeiramente, é importante notar que o objetivo da Skybound é simplificar a tarefa de gestão de recursos de companhias aéreas. Para executar corretamente esta tarefa são recomendados como especificações mínimas de hardware utilizar um processador Intel core i5-12400 ou, alternativamente, um processador AMD Ryzen 5 5600G, tendo em atenção para que o computador em questão tenha à sua disposição um mínimo de 16 GB de memória RAM DDR4 ou DDR5.

Para instalar o sistema, deve começar por obter os ficheiros Skybound.jar e databaseptda.sql. Certifique-se de seguida que o JDK está corretamente instalado e configurado no sistema. É possível verificar se o JDK está funcional ao executar o comando `java -version` no terminal da sua máquina. Para então iniciar a aplicação, basta abrir o explorador de ficheiros, navegar até o diretório onde se encontra Skybound.jar e executar o ficheiro.

Alguns problemas comuns que pode encontrar são o erro “comando java não encontrado” para o qual recomendamos primeiro confirmar se o JDK está instalado na sua máquina, assim como erro “erro de ligação à base de dados” para o qual é recomendado a verificar se o servidor da base de dados está ativo e acessível na rede. Se necessitar de suporte adicional, está livre para entrar em contacto com a nossa equipa no número xxxxxxxxx.

3 Testes e validação

[reportar estratégia de testes e respectivos resultados]

```
public class SkyBoundGestaoVoosTest {
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy/MM/dd HH:mm:ss"); no usage
    @BeforeEach
    void setUp() { Flight.getFlights().clear(); }

    @Test
    void testAdicionarVoo() {
        Flight.addFlight( id_Airplane: 1, id_Flight: 101, maxPassengers: 150, new Date(), new Date(
        assertEquals( expected: 1, Flight.getFlights().size());
    }

    @Test
    void testRemoverVoo() {
        Flight.addFlight( id_Airplane: 1, id_Flight: 101, maxPassengers: 150, new Date(), new Date(
        Flight.removeFlight( flightIndex: 0);
        assertEquals( expected: 0, Flight.getFlights().size());
    }
}
```

Testámos a classe SkyboundGestaoVoos com o objetivo de verificar se esta era capaz de adicionar e remover voos. A classe reprovou ao teste de adicionar voos e passou ao teste de remover voos, devido a complicações não previstas.

```

8
9 public class GestaoServicosClassesTest {
10     private Class testClass; 6 usages
11     private ArrayList<String> testServices; 7 usages
12
13     @BeforeEach
14     void setUp() {
15         testServices = new ArrayList<>();
16         testServices.add("Bagagem Extra");
17         testServices.add("Refeição Gourmet");
18         testClass = new Class( className: "Luxuosa", price: 200.00, seatCapacity: 10, testService
19     }
20
21     @Test
22     void testClassCreation() {
23         assertNotNull(testClass, message: "Class should be created successfully");
24         assertEquals( expected: "Luxuosa", testClass.getClassName(), message: "Class name sho
25         assertEquals( expected: 200.00, testClass.getPrice(), message: "Price should match");
26         assertEquals( expected: 10, testClass.getSeatCapacity(), message: "Seat capacity sho
27     }
28
29     @Test
30     void testServicesList() {
31         ArrayList<String> services = testClass.getServices();

```

Realizámos 5 testes para a classe GestaoServicosClasse: para testar a criação de classe, a lista de serviços, se dava erro caso um campo fosse vazio na criação de class e o mesmo para o serviço, assim como se o programa consegue registar listas de serviço vazias. A aplicação falhou nos testes de campos inválidos, de a não estarem adaptados ao estado atual do código.


```

public class CompraBilheteTest {
    void testPassengerValidation() {
        new Passenger( name: "John Doe", age: 30, email: "invalid-email", id_Passenger: 2
    }, message: "Should throw exception for invalid email format");
    }

    @Test
    void testTicketPrice() { assertTrue( condition: testTicket.getPrice() > 0, message: "Tic

    @Test
    void testDestinationValidation() {
        assertThrows(IllegalArgumentException.class, () -> {
            new Ticket( destination: "Lisboa", source: "", id_Ticket: 12345, price: 150.00);
        }, message: "Should throw exception for empty destination");
    }

    @Test
    void testSourceDestinationEquality() {
        assertThrows(IllegalArgumentException.class, () -> {
            new Ticket( destination: "Lisboa", source: "Lisboa", id_Ticket: 12345, price: 150.00)
        }, message: "Should throw exception when source and destination are the same");
    }
}

```

Realizámos 5 testes para a classe compraBilhetes. Estes englobavam casos de criação de bilhete, de validação do destino da viagem, o que aconteceria caso o destino e a origem fossem iguais, a validação do passageiro e o preço do bilhete, que tem de ser acima de zero. A aplicação passou aos testes de preço de bilhete e de validação do destino, devido a complicações não previstas.

4 Anexos

[se necessário documentação deve ser anexada e convenientemente indexada de modo a ser referenciada no documento]

[listar ficheiros externos, nomeadamente os ficheiros .mdl a consultar]