

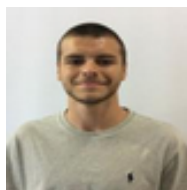
# Projeto de Desenvolvimento de Sistemas de Software - G36

Diogo Braga A82547      João Silva A82005  
Ricardo Caçador A81064      Ricardo Veloso A81919

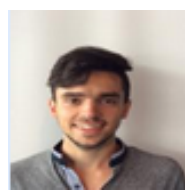
6 de Janeiro de 2019

## Resumo

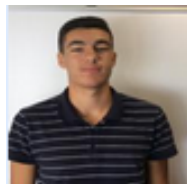
Neste relatório é apresentado um projeto baseado num configurador de carros (ConfiguraFácil), desenvolvido no âmbito da unidade curricular de Desenvolvimento de Sistemas de Software.



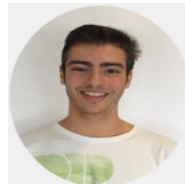
(a) Diogo Braga



(b) João Silva



(c) Ricardo Caçador



(d) Ricardo Veloso

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>5</b>
<b>2</b>	<b>Modelação</b>	<b>6</b>
2.1	Modelo de Domínio . . . . .	6
2.2	Modelo de Use Cases . . . . .	7
2.2.1	Especificação textual UC . . . . .	7
2.3	Protótipo da interface . . . . .	10
2.3.1	Página Inicial . . . . .	10
2.3.2	Admin . . . . .	10
2.3.3	Funcionário Fábrica . . . . .	11
2.3.4	Funcionário . . . . .	12
2.4	Máquina de estado da navegação . . . . .	14
2.4.1	Página Inicial . . . . .	14
2.4.2	Admin . . . . .	14
2.4.3	Funcionário Fábrica . . . . .	14
2.4.4	Funcionário . . . . .	15
2.5	Diagrama de sequência de Sistema . . . . .	16
2.6	Diagrama de sequência com subsistemas . . . . .	19
2.7	Diagrama de packages . . . . .	22
2.8	Diagrama de classe com estruturas de dados . . . . .	23
2.9	Diagrama de classe com ORM . . . . .	23
2.10	Diagrama de sequência de implementação . . . . .	24
<b>3</b>	<b>Implementação</b>	<b>29</b>
3.1	Modelo de dados . . . . .	29
3.1.1	Modelo Concetual . . . . .	29
3.1.2	Modelo Lógico . . . . .	29
3.1.3	Modelo Físico . . . . .	30
3.2	Detalhes de Implementação . . . . .	32
<b>4</b>	<b>Conclusões</b>	<b>34</b>
4.1	Análise Crítica ao Processo de Modelação . . . . .	34
4.1.1	1ª Fase . . . . .	34
4.1.2	2ª Fase . . . . .	34
4.2	Análise Crítica ao Processo de Construção da Aplicação . . . . .	35

## Lista de Figuras

2	Modelo de Domínio . . . . .	6
3	Modelo de Use Cases . . . . .	7
4	Especificação textual UC - Fazer LogIn . . . . .	7
5	Especificação textual UC - Adicionar Componente . . . . .	8
6	Especificação textual UC - Escolher Pacote Pré-definido . . . . .	8
7	Especificação textual UC - Receber Fornecimento . . . . .	8
8	Especificação textual UC - Configuração Ótima . . . . .	9
9	Interface - LogIn . . . . .	10
10	Interface - Admin: Geral . . . . .	10
11	Interface - Admin: Adicionar Funcionário . . . . .	10
12	Interface - Funcionário Fábrica: Geral . . . . .	11
13	Interface - Funcionário Fábrica: Produzir Configuração . . . . .	11
14	Interface - Funcionário Fábrica: Receber Fornecimento . . . . .	11
15	Interface - Funcionário: Geral . . . . .	12
16	Interface - Funcionário: Configuração Ótima . . . . .	12
17	Interface - Funcionário: Criar Configuração . . . . .	12
18	Interface - Funcionário: Componente Incompatível . . . . .	13
19	Interface - Funcionário: Componente Indisponível . . . . .	13
20	Interface - Funcionário: Adicionar Dados . . . . .	13
21	Máquinas de Estado - LogIn . . . . .	14
22	Máquinas de Estado - Admin: Adicionar Funcionário . . . . .	14
23	Máquinas de Estado - Funcionário Fábrica: Produzir Configuração . . . . .	14
24	Máquinas de Estado - Funcionário Fábrica: Receber Fornecimento . . . . .	15
25	Máquinas de Estado - Funcionário: Configuração Ótima . . . . .	15
26	Máquinas de Estado - Funcionário: Criar Configuração . . . . .	15
27	Máquinas de Estado - Funcionário: Adicionar Dados . . . . .	15
28	DSS - Fazer LogIn . . . . .	16
29	DSS - Adicionar Componente . . . . .	16
30	DSS - Escolher Pacote Pré-definido . . . . .	17
31	DSS - Receber Fornecimento . . . . .	18
32	DSS - Configuração Ótima . . . . .	18
33	DS Subsistemas - Fazer LogIn . . . . .	19
34	DS Subsistemas - Adicionar Componente . . . . .	19
35	DS Subsistemas - Escolher Pacote Pré-definido . . . . .	20
36	DS Subsistemas - Receber Fornecimento . . . . .	21
37	DS Subsistemas - Configuração Ótima . . . . .	21
38	Diagrama de Package . . . . .	22
39	Diagrama de Classe . . . . .	23
40	Diagrama de Classe com ORM . . . . .	23
41	DS Implementação - Fazer LogIn . . . . .	24
42	DS Implementação - Adicionar Componente . . . . .	25
43	DS Implementação - Escolher Pacote Pré-definido . . . . .	26
44	DS Implementação - Receber Fornecimento . . . . .	27
45	DS Implementação - Configuração Ótima . . . . .	28
46	Modelo Conceptual . . . . .	29
47	Modelo Lógico . . . . .	30
48	Tabela Funcionário . . . . .	30
49	Tabela Configuração . . . . .	31

50	Tabela Componente . . . . .	31
51	Tabela Incompatibilidade . . . . .	31
52	Tabela Obrigatoriedade . . . . .	31
53	Tabela Pacote . . . . .	32
54	Tabela Pacote_Componente . . . . .	32
55	Tabela Configuracao_Pacote . . . . .	32
56	Tabela Configuracao_Componente . . . . .	32
57	Estrutura multi-camada. . . . .	33

# 1 Introdução

Este trabalho tem por base a criação de uma aplicação denominada de **ConfiguraFácil**. Nesta é possível, realizar uma encomenda, por via dum orçamento estabelecido que gera uma configuração ótima, ou por via dum personalização pessoal, através dum configuração individualizada ou dum configuração com pacotes pré-definidos. Na aplicação, é também possível fazer a chegada de stock de componentes à fábrica que sustenta a aplicação.

Com a realização deste relatório, espera-se que seja possível ao leitor entender de forma esclarecedora as ideias implementadas nos modelos a seguir seccionados.

## 2 Modelação

### 2.1 Modelo de Domínio

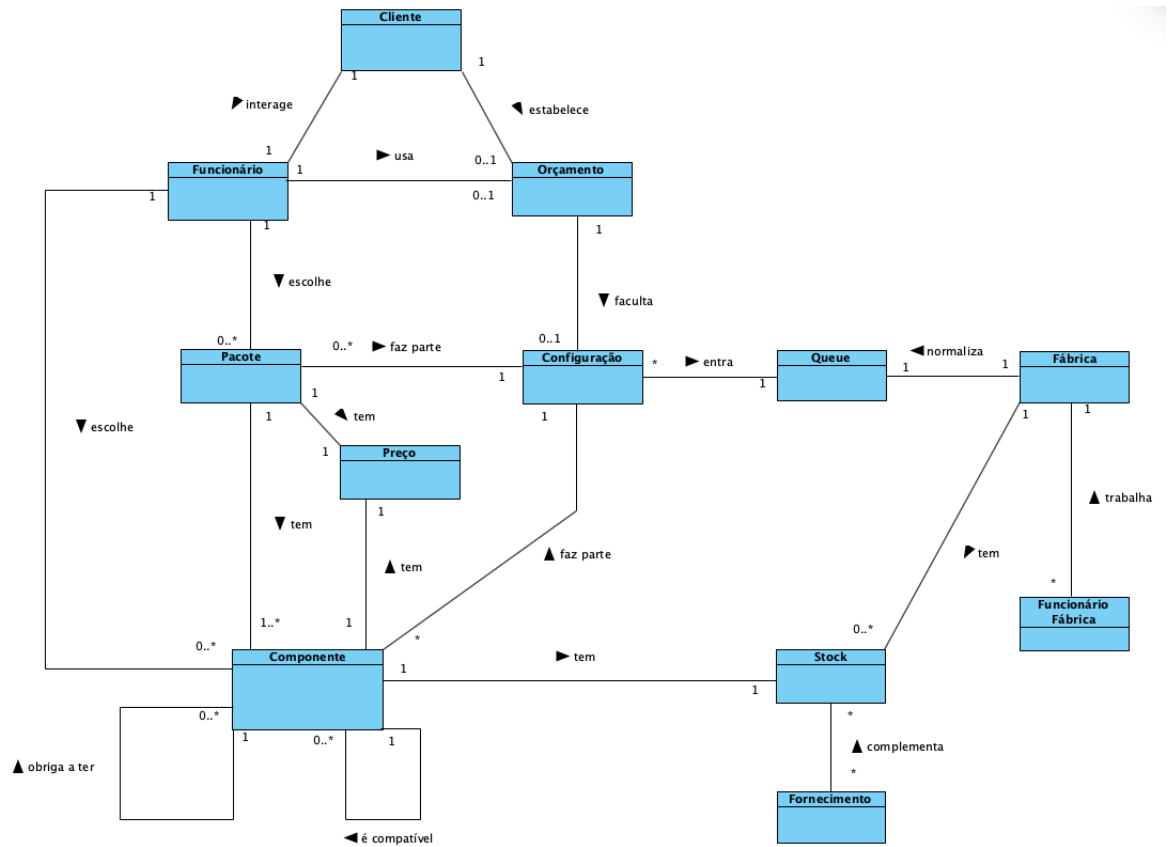


Figura 2: Modelo de Domínio

## 2.2 Modelo de Use Cases

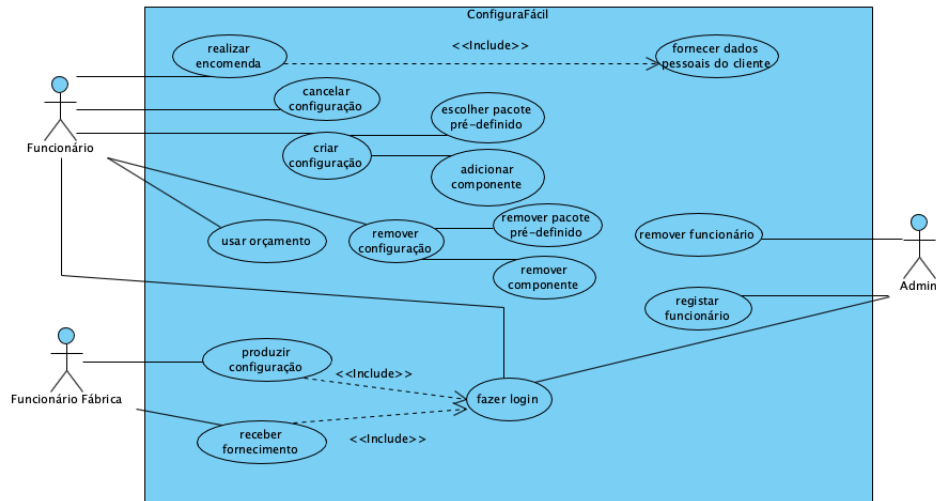


Figura 3: Modelo de Use Cases

### 2.2.1 Especificação textual UC

Use Case:	Fazer login	
Actor:	Funcionário	
Pré condição:	Ator não estar autenticado	
Pós condição:	Ator estar autenticado	
Cenário Normal	Actor input	System response
	1.Fornece dados de credenciais	2.Valida credenciais
		3.Informa ator que está autenticado
Exceção [Credenciais inválidas] (Passo 2)		2.1.Informa ator que credenciais estão incorretas

Figura 4: Especificação textual UC - Fazer LogIn

Use Case:	Adicionar Componente	
Actor:	Funcionário	
Pré condição:	Estar autenticado	
Pós condição:	Ter configuração individualizada estabelecida	
Cenário Normal	Actor input 1.<<include>> fazer login 2.Escolhe componente	System response 3.Verifica se componente está disponível 4.Verifica se existe incompatibilidades 5.Verifica se obriga a ter outros componentes 6.Associa componente à Configuração
Cenário Alternativo 1 [Existem incompatibilidades] (Passo 4)	4.3.Diz que sim	4.1.Informa das incompatibilidades
		4.2.Pergunta se quer adquirir componente com as novas compatibilidades
		4.4.Retira incompatibilidades Regressa a 5
Cenário Alternativo 2 [Existem componentes obrigatórios] (Passo 5)	5.3.Diz que sim	5.1.Informa das outras componentes necessárias
		5.2.Pergunta se pretende adquirir as novas componentes necessárias
		5.4.Adiciona todos os componentes, incluindo os novos Regressa a 6
Exceção [Componente não disponível] (Passo 3)		3.1. Informa que componente não está disponível
Exceção[Cliente diz que não] (Passo 4.3)	4.3.1 Diz que não	4.3.2. Informa que componente não vai ser aplicado
Exceção [Cliente diz que não] (Passo 5.3)	5.3.1 Diz que não	5.3.2. Informa que componente não vai ser adquirido

Figura 5: Especificação textual UC - Adicionar Componente

Use Case:	Escolher pacote pré-definido	
Actor:	Funcionário	
Pré condição:	Estar autenticado	
Pós condição:	Ter pacote escolhido	
Cenário Normal	Actor input 1.<<include>> fazer login 2.Escolhe pacote	System response 3.Verifica se pacote está disponível 4.Verifica se existem incompatibilidades 5.Verifica se existem componentes obrigatórios 6.Associa pacote ao modelo
Cenário Alternativo 1 [Existem incompatibilidades] (Passo 4)	4.3.Diz que sim	4.1.Informa das incompatibilidades
		4.2.Pergunta se quer retirar compontes incompatíveis
		4.4.Retira componentes incompatíveis 4.5.Volta ao passo 5
Cenário Alternativo 2 [Existem Componentes Obrigatórios] (Passo 5)	5.3.Diz que sim	5.1.Informa dos componentes obrigatórios
		5.2.Pergunta se quer adquirir os componentes obrigatórios
		5.4.Adiciona componentes obrigatórios 5.5.Volta ao passo 6
Exceção [Pacote não disponível] (Passo 3)		3.1. Informa que pacote não está disponível
Exceção [Cliente diz que não] (Passo 4.3)		4.3.1. Informa que pacote não vai ser aplicado
Exceção [Cliente diz que não] (Passo 5.3)		5.3.1. Informa que pacote não vai ser aplicado

Figura 6: Especificação textual UC - Escolher Pacote Pré-definido

Use Case:	Receber fornecimento	
Actor:	Funcionário Fábrica	
Pré condição:	Estar autenticado na aplicação	
Pós condição:	Stock Atualizado	
Cenário Normal	Actor input 1.<<include>> fazer login 2.Fornece código do componente  6.Fornece quantidade a adicionar ao Stock do componente	System response 3. Verifica existência do código 4. Verifica a existência do componente na fábrica 5.Pede quantidade do componente a acrescentar  7.Atualiza quantidade existente do componente em stock
Exceção [Código não existente] (Passo 3)		1.Informa que código está errado
Exceção [Componente não existe na fábrica] (Passo 4)		1. Informa que o componente não existe na fábrica

Figura 7: Especificação textual UC - Receber Fornecimento

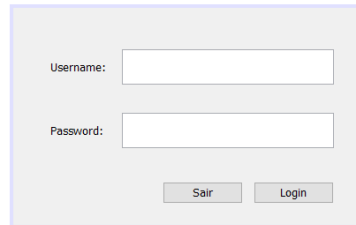


Use Case:	Configuração Ótima	
Actor:	Funcionário	
Pré condição:	Estar autenticado	
Pós condição:	Ter a configuração ótima	
Cenário Normal	Actor input	System response
	1.<<include>> fazer login	
	2.Insere orçamento	
		3.Verifica se orçamento é suficiente
		4.Gera configuração ótima
Exceção [Orçamento Insuficiente] (Passo 3)		5.Informa da configuração
		3.1. Informa que orçamento é insuficiente

Figura 8: Especificação textual UC - Configuração Ótima

## 2.3 Protótipo da interface

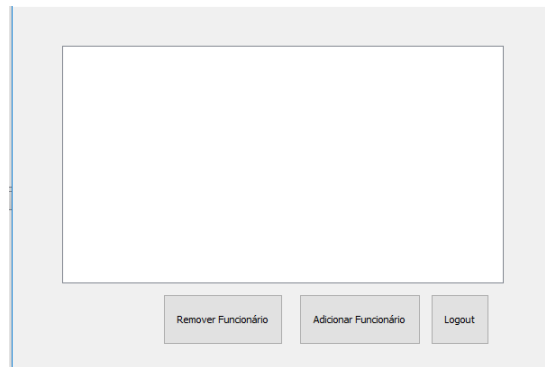
### 2.3.1 Página Inicial



A login interface prototype with a light gray background. It features two text input fields: the first is labeled "Username:" and the second is labeled "Password:". Below the password field are two buttons: "Sair" (left) and "Login" (right).


Figura 9: Interface - LogIn

### 2.3.2 Admin



An admin interface prototype with a light gray background. It contains a large, empty rectangular area in the center. At the bottom, there are three buttons: "Remover Funcionário" (left), "Adicionar Funcionário" (middle), and "Logout" (right).

Figura 10: Interface - Admin: Geral



An admin interface prototype for adding a new employee. It has a light gray background and contains five text input fields stacked vertically, labeled "Nome:", "Morada:", "Telefone:", "Salário:", and "Password:". At the bottom, there are two buttons: "Adicionar" (left) and "Voltar" (right).

Figura 11: Interface - Admin: Adicionar Funcionário

### 2.3.3 Funcionário Fábrica

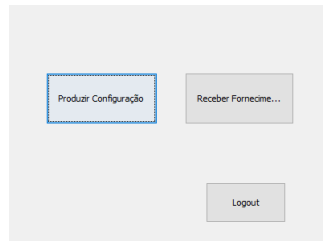


Figura 12: Interface - Funcionário Fábrica: Geral

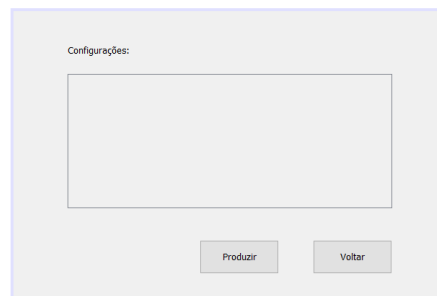


Figura 13: Interface - Funcionário Fábrica: Produzir Configuração

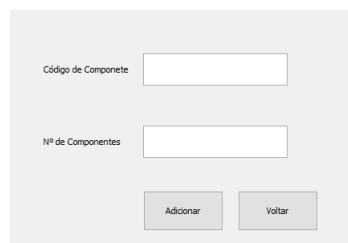


Figura 14: Interface - Funcionário Fábrica: Receber Fornecimento

#### 2.3.4 Funcionário

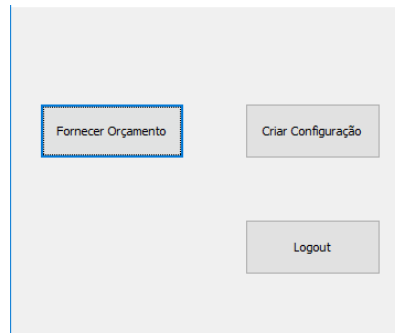


Figura 15: Interface - Funcionário: Geral

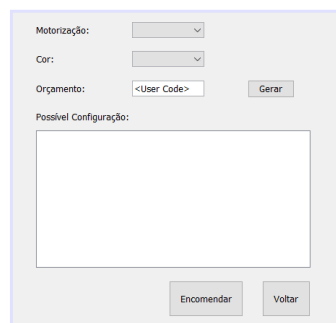


Figura 16: Interface - Funcionário: Configuração Ótima

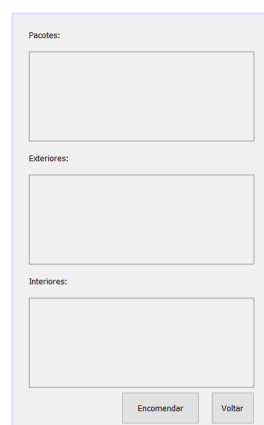
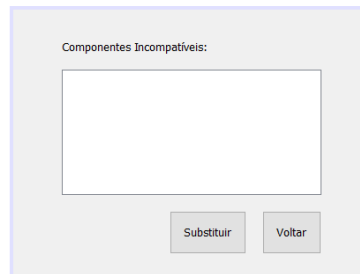


Figura 17: Interface - Funcionário: Criar Configuração



Componentes Incompatíveis:

A rectangular box for listing incompatible components.

Substituir Voltar

Figura 18: Interface - Funcionário: Componente Incompatível

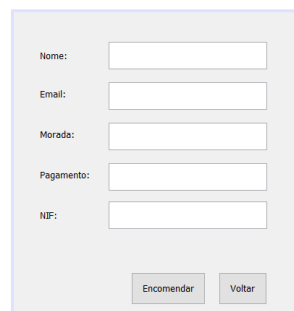
---



Componente Indisponível

OK

Figura 19: Interface - Funcionário: Componente Indisponível



Nome:

Email:

Morada:

Pagamento:

NIF:

Encomendar Voltar

Figura 20: Interface - Funcionário: Adicionar Dados

## 2.4 Máquina de estado da navegação

### 2.4.1 Página Inicial

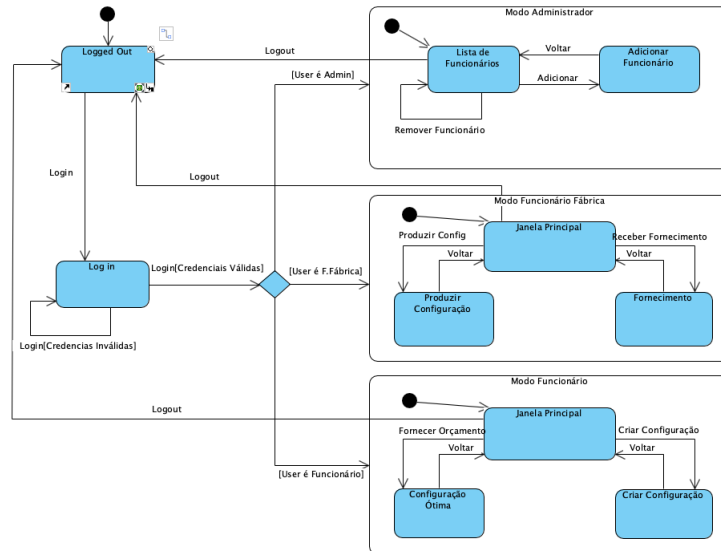


Figura 21: Máquinas de Estado - LogIn

### 2.4.2 Admin

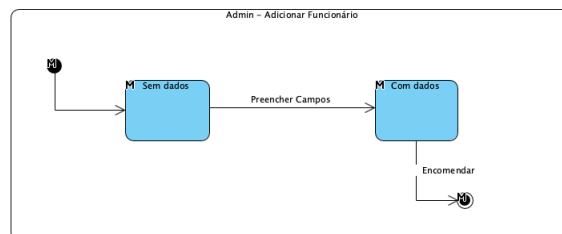


Figura 22: Máquinas de Estado - Admin: Adicionar Funcionário

### 2.4.3 Funcionário Fábrica

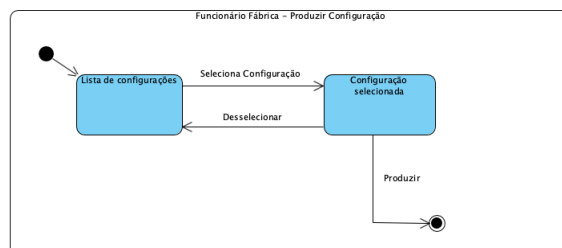


Figura 23: Máquinas de Estado - Funcionário Fábrica: Produzir Configuração

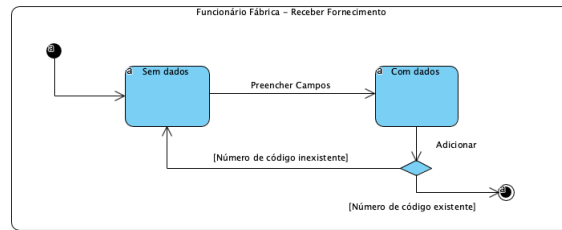


Figura 24: Máquinas de Estado - Funcionário Fábrica: Receber Fornecimento

#### 2.4.4 Funcionário

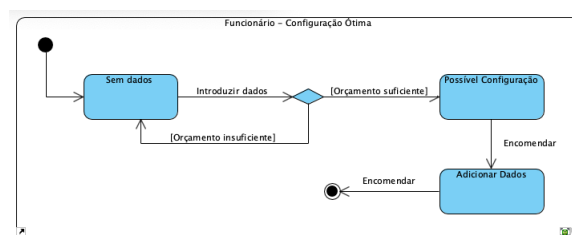


Figura 25: Máquinas de Estado - Funcionário: Configuração Ótima

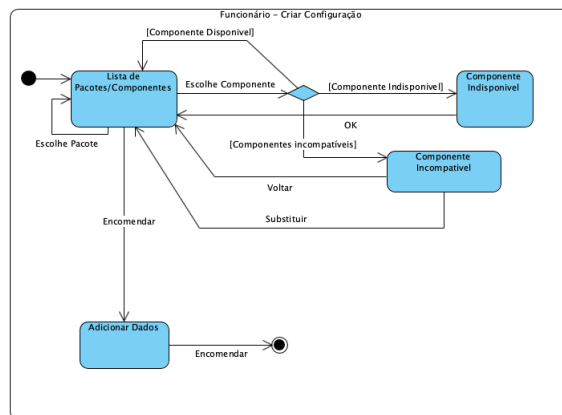


Figura 26: Máquinas de Estado - Funcionário: Criar Configuração

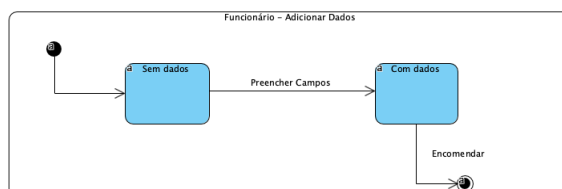


Figura 27: Máquinas de Estado - Funcionário: Adicionar Dados

## 2.5 Diagrama de sequência de Sistema

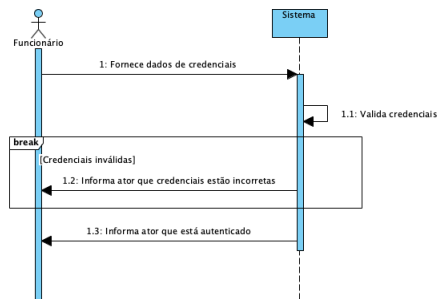


Figura 28: DSS - Fazer LogIn

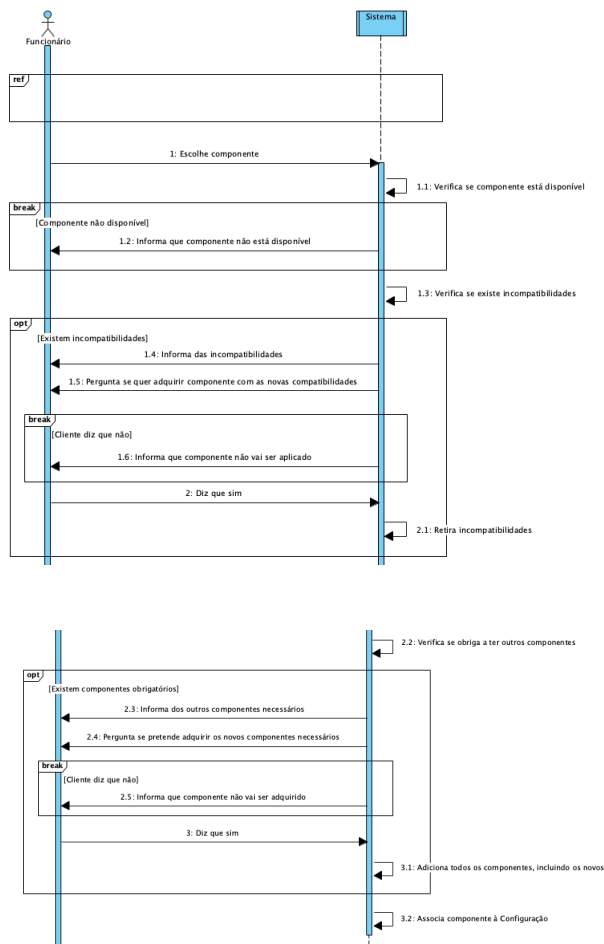


Figura 29: DSS - Adicionar Componente



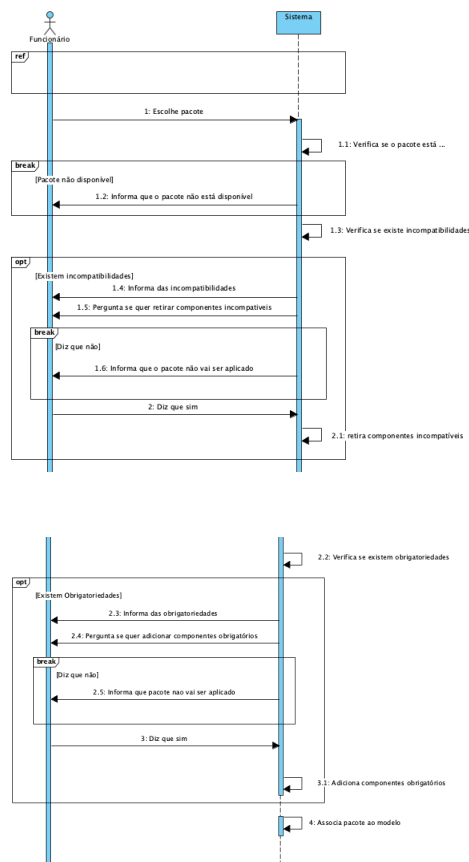


Figura 30: DSS - Escolher Pacote Pré-definido

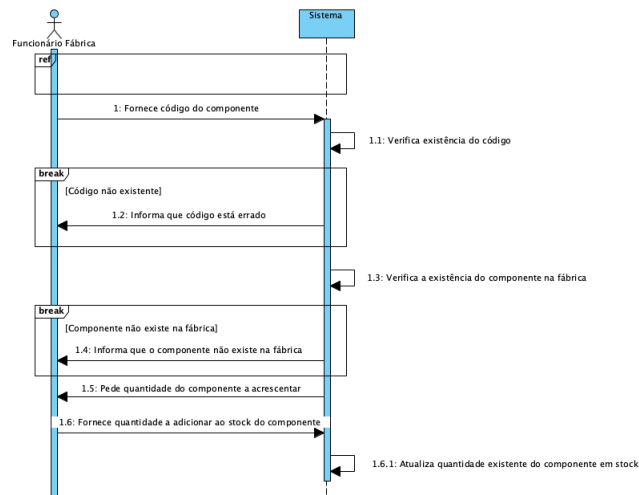


Figura 31: DSS - Receber Fornecimento

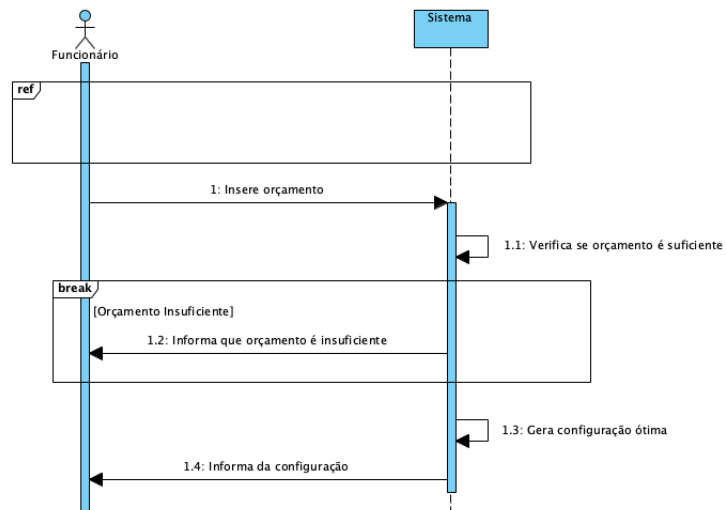


Figura 32: DSS - Configuração Ótima

## 2.6 Diagrama de sequência com subsistemas

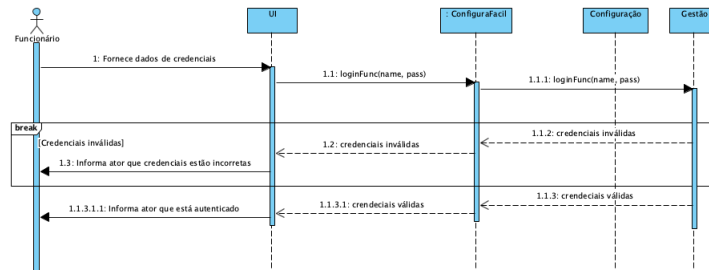


Figura 33: DS Subsistemas - Fazer LogIn

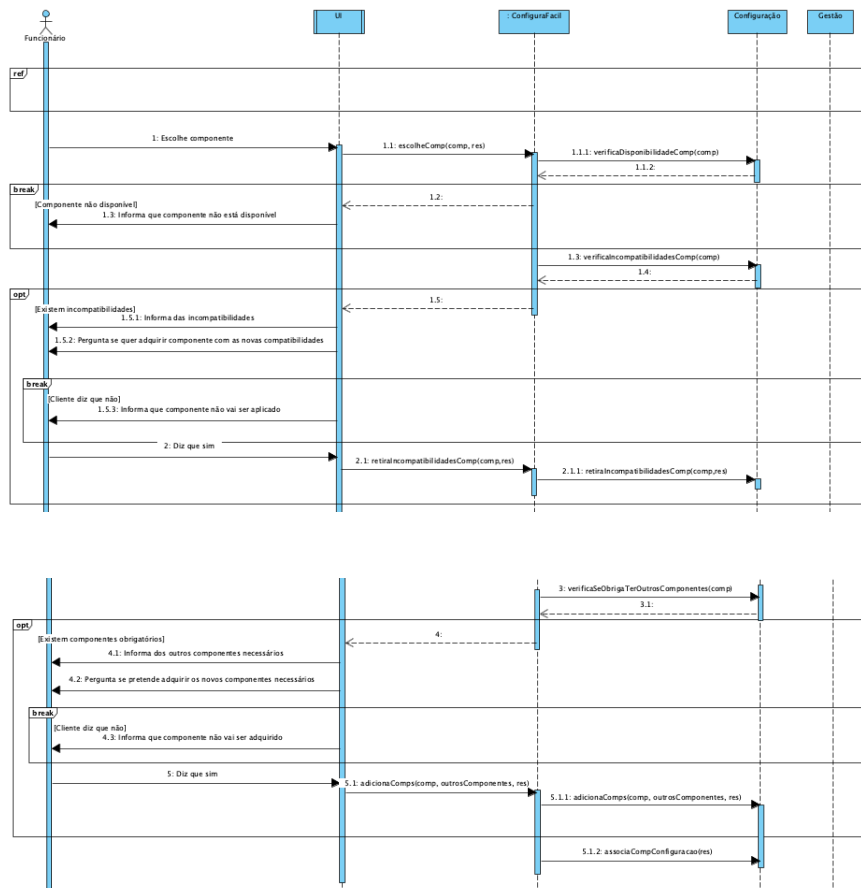


Figura 34: DS Subsistemas - Adicionar Componente

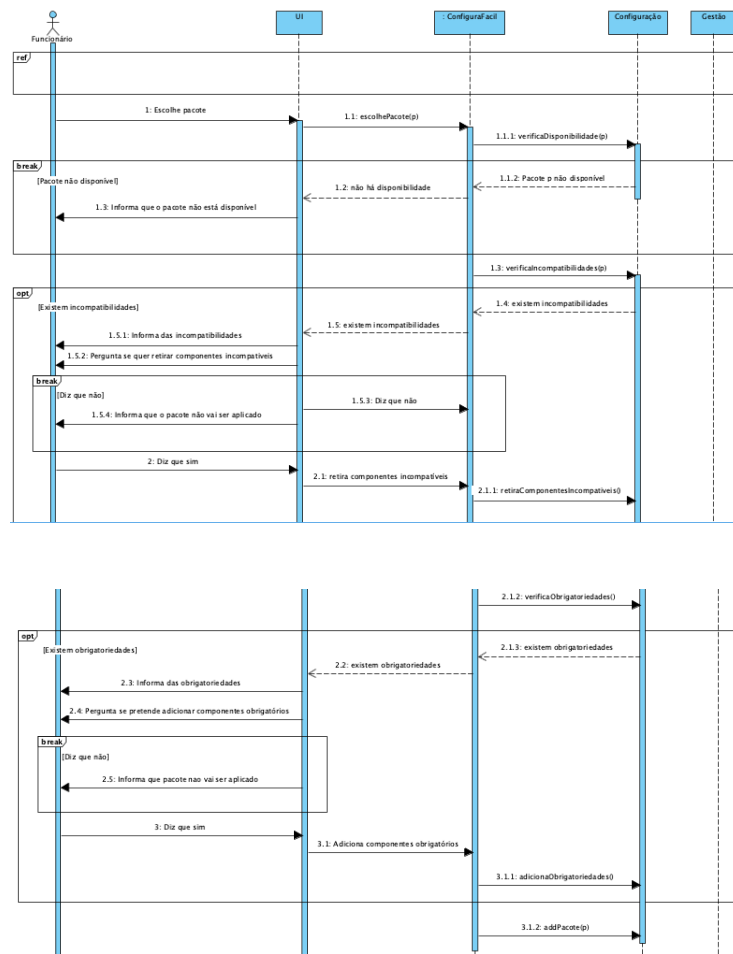


Figura 35: DS Subsistemas - Escolher Pacote Pré-definido

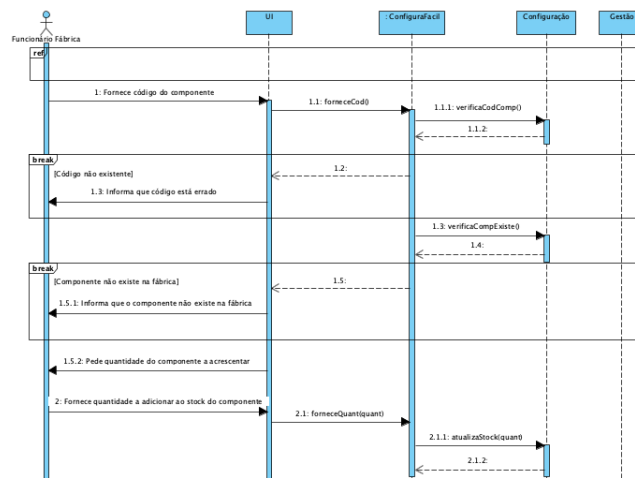


Figura 36: DS Subsistemas - Receber Fornecimento

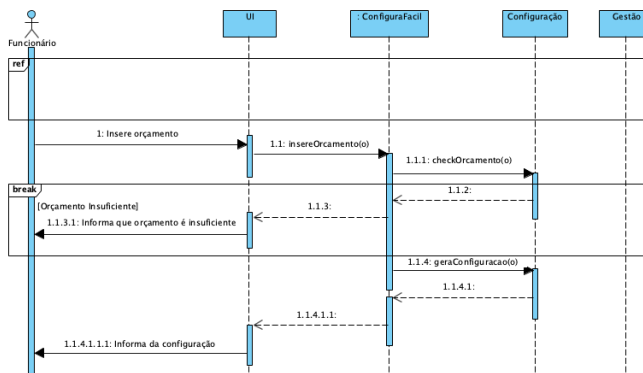


Figura 37: DS Subsistemas - Configuração Ótima

## 2.7 Diagrama de packages

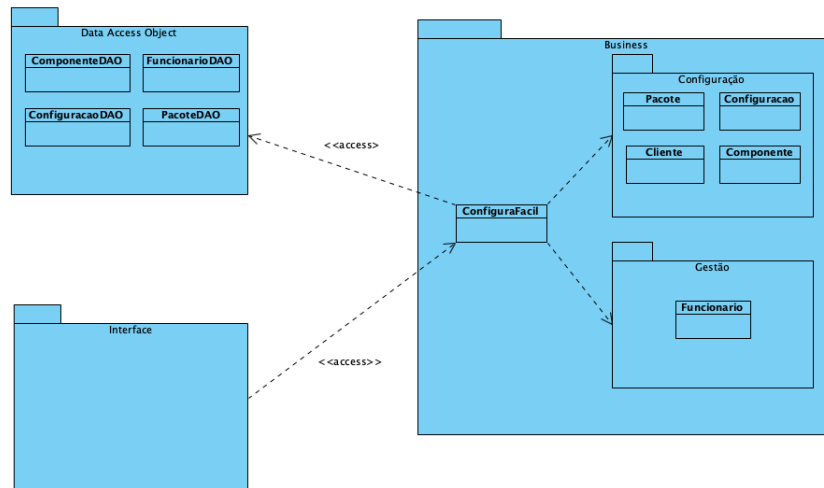


Figura 38: Diagrama de Package

## 2.8 Diagrama de classe com estruturas de dados

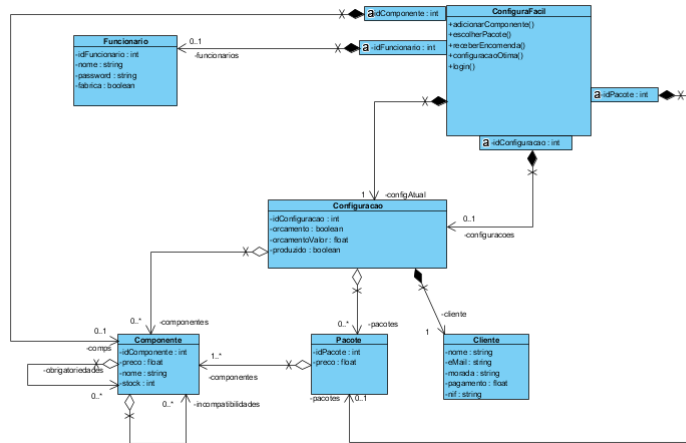


Figura 39: Diagrama de Classe

## 2.9 Diagrama de classe com ORM

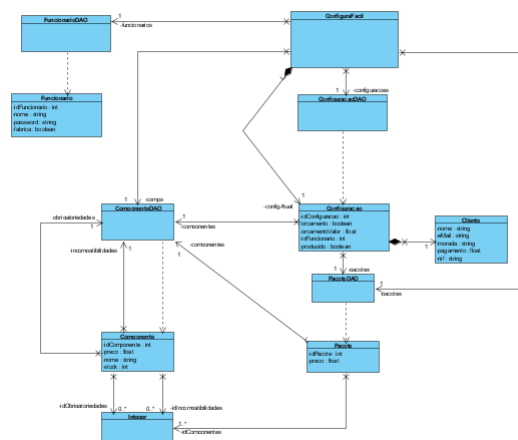


Figura 40: Diagrama de Classe com ORM

## 2.10 Diagrama de sequência de implementação

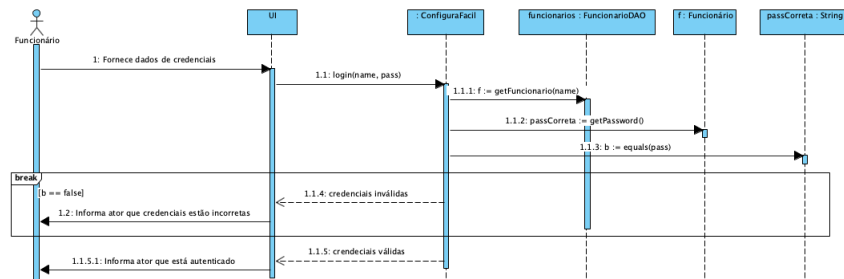
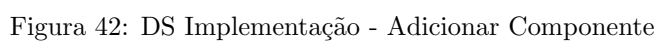


Figura 41: DS Implementação - Fazer LogIn





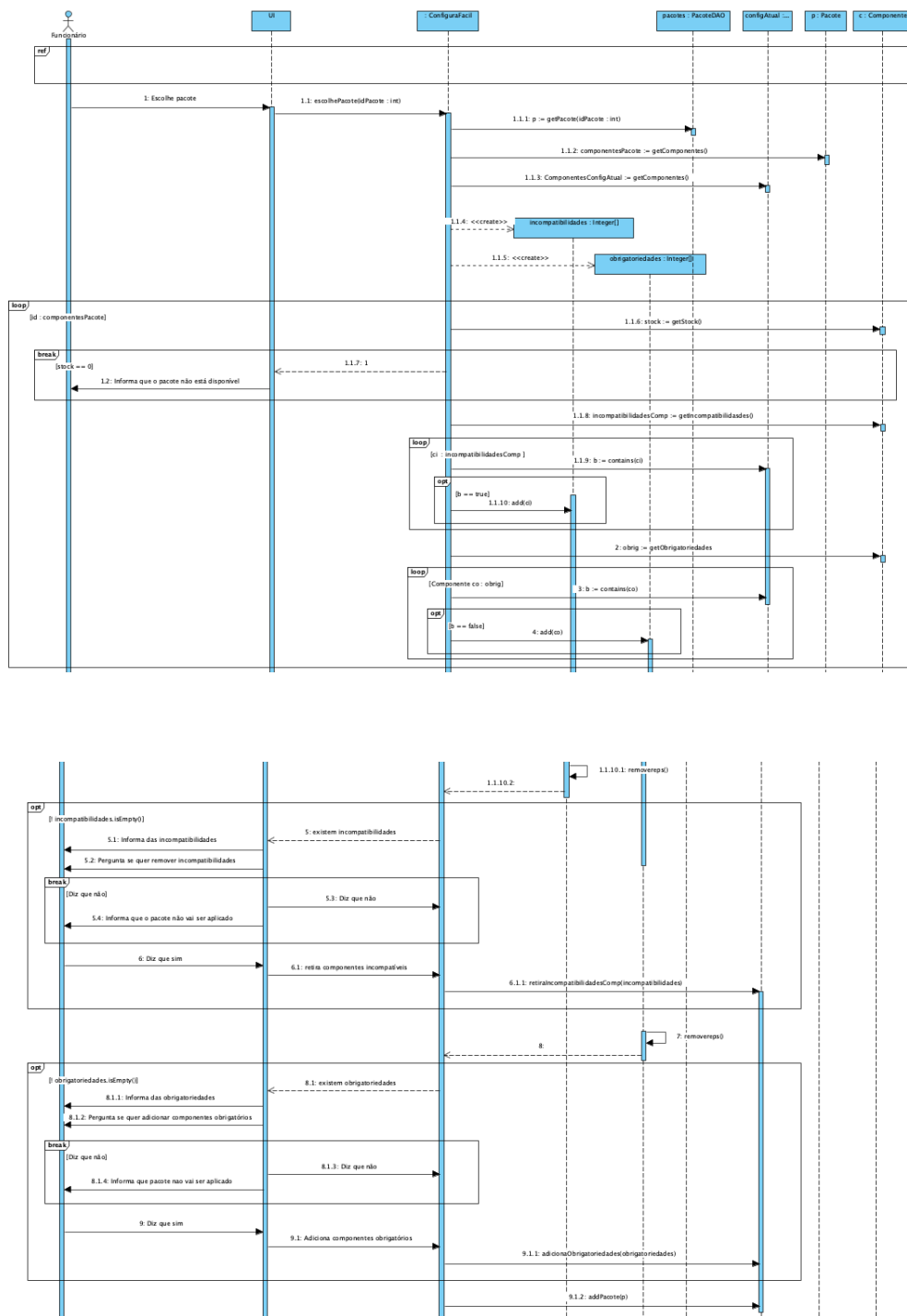


Figura 43: DS Implementação - Escolher Pacote Pré-definido

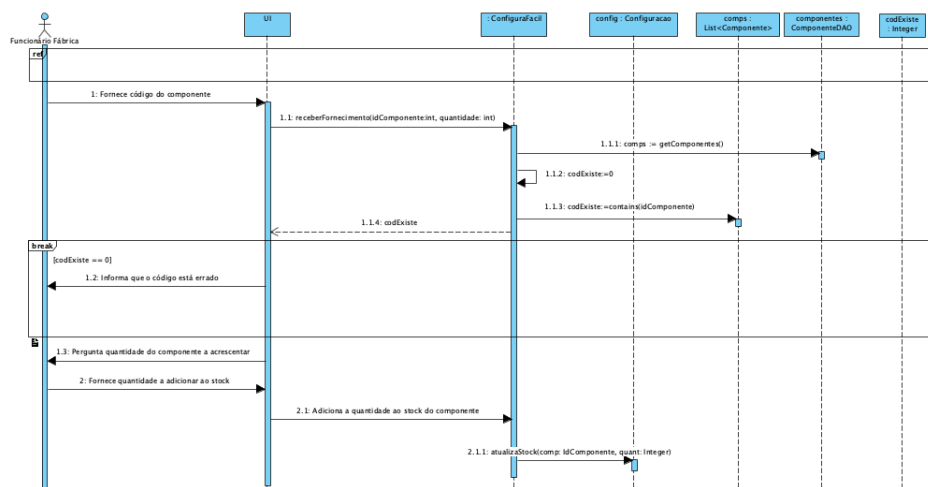


Figura 44: DS Implementação - Receber Fornecimento

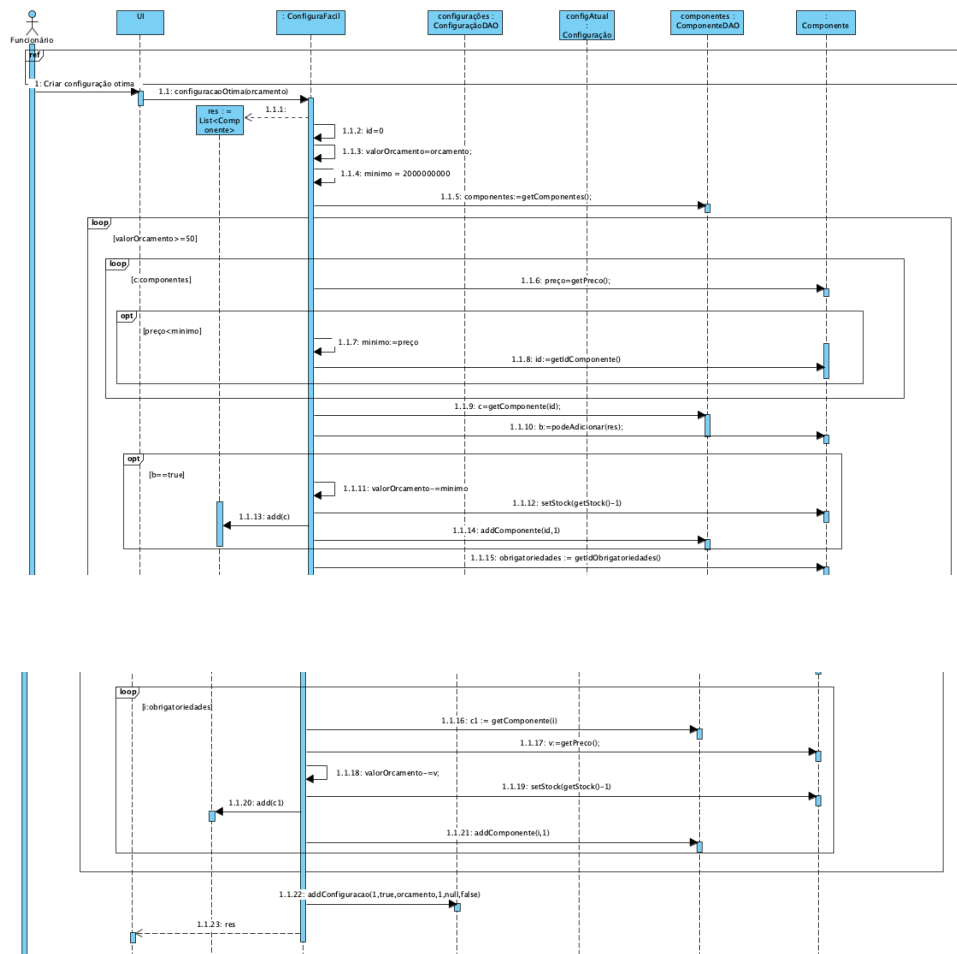


Figura 45: DS Implementação - Configuração Ótima

## 3 Implementação

### 3.1 Modelo de dados

Como suporte à aplicação concebida criamos uma Base de Dados com o único propósito de guardar a informação útil para o bom funcionamento da aplicação. Desta forma a Base de Dados é usada somente para armazenar e consultar informação.

As etapas de construção da Base de Dados são especificadas a seguir nesta secção.

#### 3.1.1 Modelo Concetual

Inicialmente começamos por identificar o mínimo de entidades possíveis para suportar os dados que a aplicação iria necessitar, bem como todos os seus atributos e relacionamentos entre cada uma das entidades. O modelo criado é apresentado na figura 46.

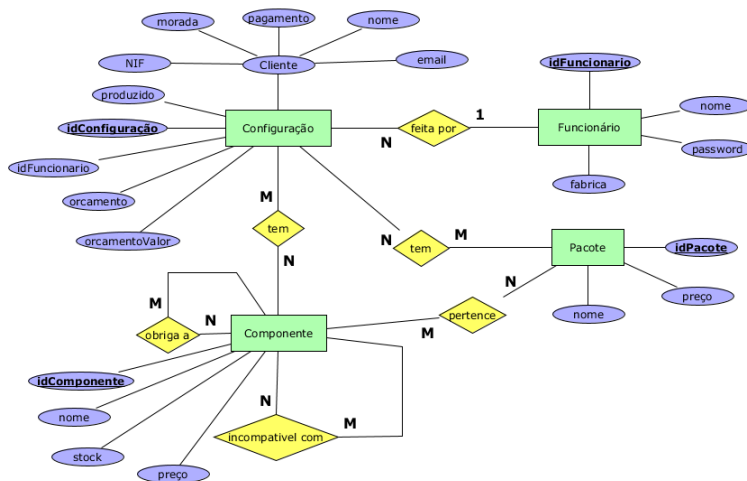


Figura 46: Modelo Conceptual

#### 3.1.2 Modelo Lógico

Transitando do modelo Concetual para o modelo Lógico, criamos o seguinte modelo apresentado na figura 47. Com este modelo temos um modelo já muito próximo da possível implementação física ou real da Base de Dados, com bastante detalhe de cada relação.

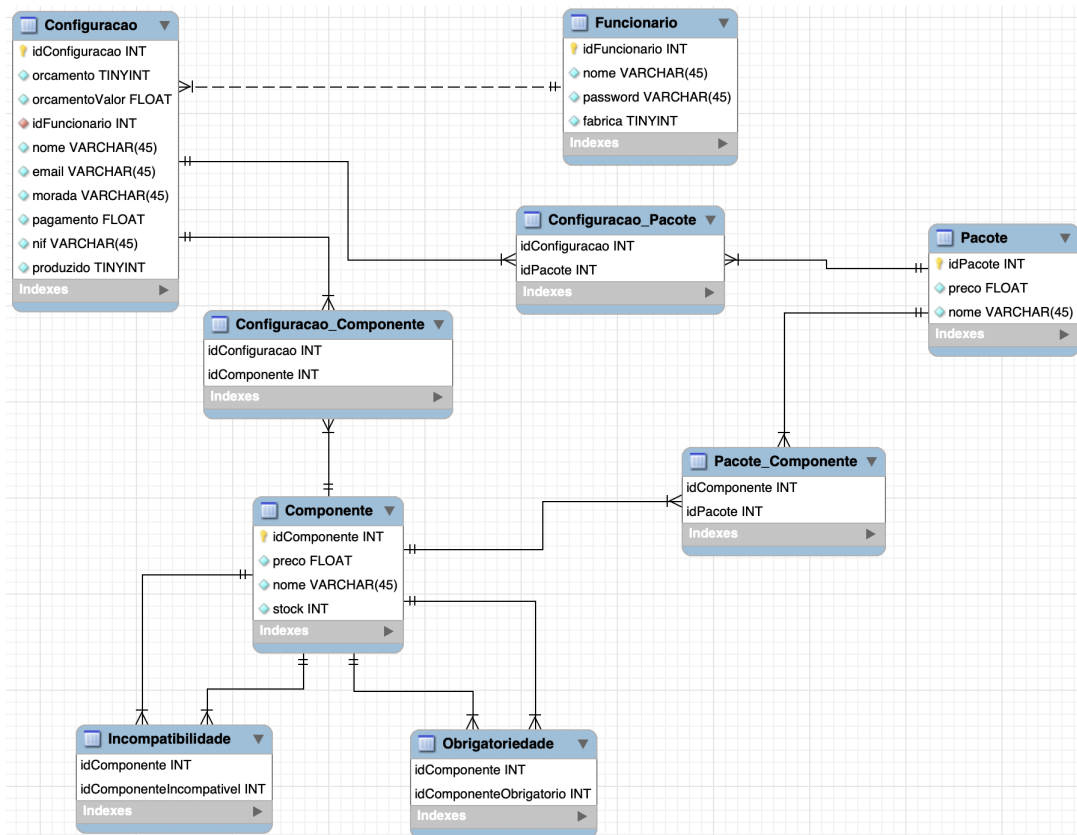


Figura 47: Modelo Lógico

### 3.1.3 Modelo Físico

Por último, nas seguintes imagens é apresentado o modelo físico, com todas as especificações e restrições necessárias para o normal funcionamento da Base de Dados.

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Funcionario` (
  `idFuncionario` INT NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `fabrica` TINYINT NOT NULL,
  PRIMARY KEY (`idFuncionario`))
ENGINE = InnoDB;
```

Figura 48: Tabela Funcionário

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Configuracao` (
  `idConfiguracao` INT NOT NULL AUTO_INCREMENT,
  `orcamento` TINYINT NOT NULL,
  `orcamentoValor` FLOAT NOT NULL,
  `idFuncionario` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `morada` VARCHAR(45) NOT NULL,
  `pagamento` FLOAT NOT NULL,
  `nif` VARCHAR(45) NOT NULL,
  `produzido` TINYINT NOT NULL,
  PRIMARY KEY (`idConfiguracao`),
  INDEX `fk_Configuracao_Funcionario1_idx` (`idFuncionario` ASC) VISIBLE,
  CONSTRAINT `fk_Configuracao_Funcionario1`
    FOREIGN KEY (`idFuncionario`)
      REFERENCES `ConfiguraFacil`.`Funcionario` (`idFuncionario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 49: Tabela Configuração

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Componente` (
  `idComponente` INT NOT NULL AUTO_INCREMENT,
  `preco` FLOAT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `stock` INT NOT NULL,
  PRIMARY KEY (`idComponente`))
ENGINE = InnoDB;
```

Figura 50: Tabela Componente

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Incompatibilidade` (
  `idComponente` INT NOT NULL,
  `idComponenteIncompativel` INT NOT NULL,
  PRIMARY KEY (`idComponente`, `idComponenteIncompativel`),
  INDEX `fk_Componente_has_Componente_Componente2_idx` (`idComponenteIncompativel` ASC) VISIBLE,
  INDEX `fk_Componente_has_Componente_Componente1_idx` (`idComponente` ASC) VISIBLE,
  CONSTRAINT `fk_Componente_has_Componente_Componente1`
    FOREIGN KEY (`idComponente`)
      REFERENCES `ConfiguraFacil`.`Componente` (`idComponente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Componente_has_Componente_Componente2`
    FOREIGN KEY (`idComponenteIncompativel`)
      REFERENCES `ConfiguraFacil`.`Componente` (`idComponente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 51: Tabela Incompatibilidade

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Obrigatoriedade` (
  `idComponente` INT NOT NULL,
  `idComponenteObrigatorio` INT NOT NULL,
  PRIMARY KEY (`idComponente`, `idComponenteObrigatorio`),
  INDEX `fk_Componente_has_Componente_Componente4_idx` (`idComponenteObrigatorio` ASC) VISIBLE,
  INDEX `fk_Componente_has_Componente_Componente3_idx` (`idComponente` ASC) VISIBLE,
  CONSTRAINT `fk_Componente_has_Componente_Componente3`
    FOREIGN KEY (`idComponente`)
      REFERENCES `ConfiguraFacil`.`Componente` (`idComponente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Componente_has_Componente_Componente4`
    FOREIGN KEY (`idComponenteObrigatorio`)
      REFERENCES `ConfiguraFacil`.`Componente` (`idComponente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 52: Tabela Obrigatoriedade

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Pacote` (
  `idPacote` INT NOT NULL,
  `preco` FLOAT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idPacote`))
ENGINE = InnoDB;
```

Figura 53: Tabela Pacote

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Pacote_Componente` (
  `idComponente` INT NOT NULL AUTO_INCREMENT,
  `idPacote` INT NOT NULL,
  PRIMARY KEY (`idComponente`, `idPacote`),
  INDEX `fk_Componente_has_Pacote_Pacote1_idx` (`idPacote` ASC) VISIBLE,
  INDEX `fk_Componente_has_Pacote_Componente1_idx` (`idComponente` ASC) VISIBLE,
  CONSTRAINT `fk_Componente_has_Pacote_Componente1`
    FOREIGN KEY (`idComponente`)
      REFERENCES `ConfiguraFacil`.`Componente` (`idComponente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Componente_has_Pacote_Pacote1`
    FOREIGN KEY (`idPacote`)
      REFERENCES `ConfiguraFacil`.`Pacote` (`idPacote`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 54: Tabela Pacote\_Componente

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Configuracao_Pacote` (
  `idConfiguracao` INT NOT NULL,
  `idPacote` INT NOT NULL,
  PRIMARY KEY (`idConfiguracao`, `idPacote`),
  INDEX `fk_Configuracao_has_Pacote_Pacote1_idx` (`idPacote` ASC) VISIBLE,
  INDEX `fk_Configuracao_has_Pacote_Configuracao1_idx` (`idConfiguracao` ASC) VISIBLE,
  CONSTRAINT `fk_Configuracao_has_Pacote_Configuracao1`
    FOREIGN KEY (`idConfiguracao`)
      REFERENCES `ConfiguraFacil`.`Configuracao` (`idConfiguracao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Configuracao_has_Pacote_Pacote1`
    FOREIGN KEY (`idPacote`)
      REFERENCES `ConfiguraFacil`.`Pacote` (`idPacote`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 55: Tabela Configuracao\_Pacote

```
CREATE TABLE IF NOT EXISTS `ConfiguraFacil`.`Configuracao_Componente` (
  `idConfiguracao` INT NOT NULL,
  `idComponente` INT NOT NULL,
  PRIMARY KEY (`idConfiguracao`, `idComponente`),
  INDEX `fk_Configuracao_has_Componente_Componente1_idx` (`idComponente` ASC) VISIBLE,
  INDEX `fk_Configuracao_has_Componente_Configuracao1_idx` (`idConfiguracao` ASC) VISIBLE,
  CONSTRAINT `fk_Configuracao_has_Componente_Configuracao1`
    FOREIGN KEY (`idConfiguracao`)
      REFERENCES `ConfiguraFacil`.`Configuracao` (`idConfiguracao`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Configuracao_has_Componente_Componente1`
    FOREIGN KEY (`idComponente`)
      REFERENCES `ConfiguraFacil`.`Componente` (`idComponente`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Figura 56: Tabela Configuracao\_Componente

## 3.2 Detalhes de Implementação

Nesta fase do trabalho, focamo-nos na transição da parte da modelação do projeto para a parte mais física, implementando assim o processo descrito



anteriormente.

Para a realização desta transição baseamo-nos obviamente em todo o trabalho de modelação que já tinha sido feito. Começando pela estruturação de ficheiros na arquitetura multi-camada desenvolvida apoiamo-nos no respetivo **diagrama de packages** chegando à estrutura descrita na imagem 57.

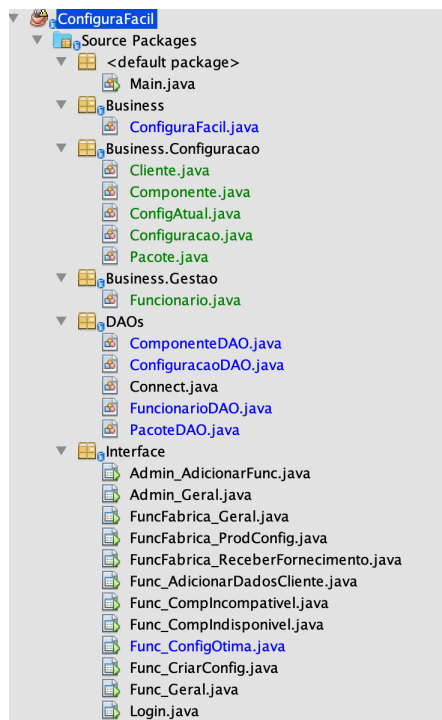


Figura 57: Estrutura multi-camada.

As classes que a nossa implementação contém são referentes às classes que referimos nos diagramas de classe, como possível verificar na mesma imagem.

Quanto aos Use Cases que é necessário que sejam implementados, estes encontram-se no FACADE da camada de negócio que é respetivamente a ConfiguraFacil. As suas implementações respeitam em grande parte os diagramas de sequência de implementação apresentados, apenas pequenas modificações foram feitas por forma a melhorar o código.

Os packages em que o projeto está dividido é o que torna a arquitetura num modelo multi-camada. Isto porque a parte da Interface só comunica com a camada de negócio estritamente pelo FACADE, ConfiguraFacil.

## 4 Conclusões

### 4.1 Análise Crítica ao Processo de Modelação

#### 4.1.1 1ª Fase

Nesta primeira fase, começamos por analisar os requisitos do sistema, presentes no enunciado e recebidos diretamente dos stakeholders (professores), de forma a modelar a estrutura inicial do sistema como um todo da forma mais simplificada possível, ficando assim com uma base definida para o trabalho futuro nos seguintes diagramas.

O Modelo de Domínio sofreu imensas alterações durante a elaboração do trabalho, muito devido à necessidade da criação de classes, para que este consiga responder a todas as funcionalidades presentes no Diagrama de Use Cases. Após todas estas alterações podemos afirmar que estamos bastante satisfeitos com a apresentação final do Modelo de Domínio.

No Diagrama de Use Cases o mesmo se verificou ao nível de alterações. Inicialmente o diagrama estava bastante básico, ainda que já contivesse todos os Use Cases que se consideravam indispensáveis. As mudanças neste diagrama visaram a utilização de funcionalidades mais avançadas (como includes) e a adição de alguns Use Cases que achamos relevantes para o bom funcionamento do Software - ConfiguraFácil.

Na especificação dos três principais Use Cases tivemos atenção a todos os cenários alternativos e de exceção e estamos bastante satisfeitos com a qualidade apresentada.

Optámos por desenhar primeiro a interface antes de definir o Diagrama de Máquinas de Estado, isto porque este se debruça sobre as janelas da interface. O desenho da interface foi planeado de forma a que todos os utilizadores do Software tenham facilidade ao navegar por este mantendo todas as funcionalidades disponíveis. A partir deste momento o desenvolvimento das Máquinas de Estado teve como guião a interface o que facilitou bastante o mesmo. O Diagrama poderá ser alvo de melhorias futuramente mas, de momento, consideramos que se encontra bem definido.

#### 4.1.2 2ª Fase

Na segunda fase, começamos por desenhar os DS (Diagramas de Sequência) de Sistema e os do UI/Facade. De seguida foram desenhados os Diagramas de Classe, com e sem DAOs. Por fim foram desenhados os DS de Subsistemas e os de Implementação. Vamos passar a explicar cada sub fase desta 2ª fase e o porquê desta ordem em específico.

O desenho dos DSS foram extremamente importantes no que conta ao aperfeiçoamento da 1ª Fase, temos melhorando consideravelmente a especificação dos Use Cases em formato tabular. A partir daí o processo foi simples e estamos bastante satisfeitos com a qualidade dos DSS.

Tal como no desenho dos DSS, o desenho dos DS do UI/Facade foi igualmente simples sendo que a variação de um DS para o outro não é consideravelmente grande ou complicada.

Os Diagramas de Classe (tanto com DAOs como sem DAOs) foram bastante ponderados entre o grupo isto para que a implementação da aplicação estivesse de acordo com aquilo que tinha sido definido. Foram muitas as alterações ao

longo do projeto de forma a simplificar os Diagramas sem lhes tirar qualquer tipo de funcionalidade necessária. Concluimos que estes foram bem conseguidos e apresentam todas as funcionalidades necessárias para o bom funcionamento da aplicação e constituem uma boa ferramenta para o desenho dos DS de Subsistemas e Implementação.

Para os DS de Subsistemas concluimos que era boa prática definir 2 subsistemas : Gestão e Configuração. O subsistema de Gestão serve para realizar tarefas de autenticação e qualquer tipo de funcionalidade praticada pelo Admin. O subsistema de Configuração serve para realizar praticamente todas as outras tarefas impostas no projeto tais como : Verificar Incompatibilidades, Criar Configurações, Adicionar Componentes a uma Configuração. O desenho destes DS também foi um processo rápido pois, utilizando os DS anteriores, foi simples estabelecer as mensagens que estavam destinadas a cada Subsistema em específico.

Os do DS de Implementação foram, claramente, os mais complicados de implementar. Devido há enorme quantidade de cenários de exceção e alternativos juntamente com a utilização de ciclos (loops) para que seja possível transparecer cada DS para as suas respetivas funções em Java tornou-se algo complicado desenhar todos eles, ainda que uns mais que outros. Fora estas dificuldades concluimos que estes DS estão bem definidos ainda que futuramente podessem ser alvos de melhoria.

## **4.2 Análise Crítica ao Processo de Construção da Aplicação**

A comunicação entre a aplicação e a base de dados criada não foi muito bem conseguida, sendo que as únicas funcionalidades operacionais são as de receber encomenda e fazer login.

A UI está de acordo com os protótipos realizados e com as máquinas de estado desenvolvidas.