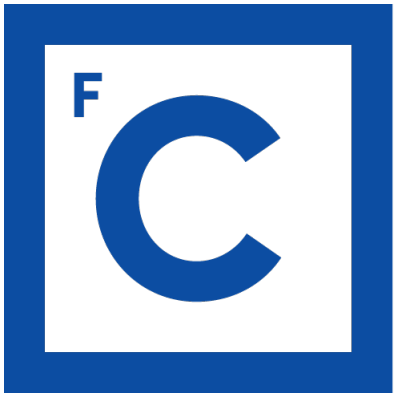

Ciências ULisboa

Universidade de Lisboa

Mestrado em Engenharia Informática

Design de Software 2018/2019

Projeto 2 - Dispositivos IoT

Autores:
Paulo Santos 47806
Rafael Melo 47878
Diogo Pereira 47888
Professora:
Prof. Antónia Lopes

30 de Dezembro de 2018

Conteúdo

1 Funcionalidades e Produtos **2**
1.1 Feature Model 2
1.2 Tabela de Produtos 2

2 Arquitetura **3**
2.1 Noções sobre a Arquitetura do Sistema 3
2.2 Vista de Módulos 4
2.3 Vista de Componentes e Conectores 6
2.3.1 Vista C&C do Monitor de Movimento 6
2.3.2 Vista C&C do Monitor de Ambiente 6
2.3.3 Vista C&C do Output 7
2.3.4 Vista C&C do Monitor de Qualidade do Ar 8
2.3.5 Vista C&C do Dispositivo Vestível 8
2.3.6 Vista C&C da Linguagem 9

3 Decisões de Implementação **10**
3.1 Decisões tomadas na Implementação 10
3.2 Decisões relativas à visualização do Output 11

Capítulo 1

Funcionalidades e Produtos

1.1 Feature Model

De acordo com os requisitos descritos no enunciado deste projeto realizámos o seguinte modelo de funcionalidades:

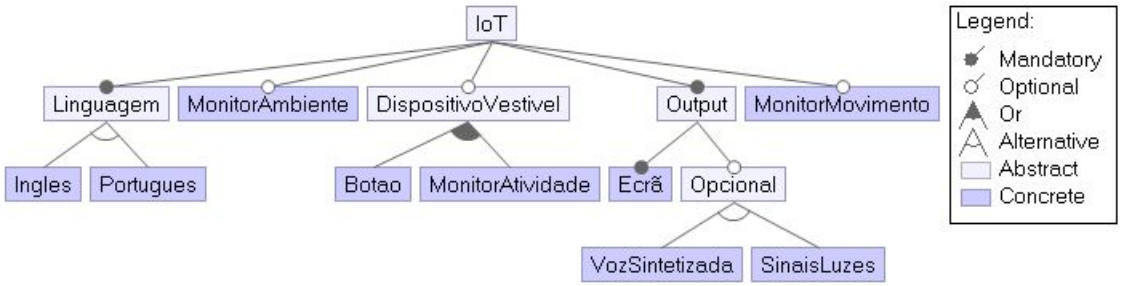

Figura 1.1: Modelo de Funcionalidades

A nossa família de produtos implementa obrigatoriamente um tipo de linguagem que pode ser inglesa ou portuguesa, um monitor de ambiente opcional, um dispositivo vestível que apesar de ser opcional quando presente manifesta-se num dispositivo com botão, num monitor de atividade, ou ambos.

O output é obrigatório e exibe-se sempre no ecrã, sendo que pode ser complementada com sinais de luzes, para utilizadores surdos, ou voz sintetizada, para utilizadores invisuais. Por fim o monitor de movimento é opcional e é uma funcionalidade extra.

1.2 Tabela de Produtos

Com base no modelo de funcionalidades apresentado anteriormente decidiu-se construir oito produtos diferentes que implementam funcionalidades dispares. Encontram-se em destaque os quatro produtos que foram escolhidos para realizar a implementação.

	Linguagem		Monitor de Ambiente	Monitor de Qualidade do Ar	Monitor de Movimento	Dispositivo Vestível		Output	
	Inglês	Português				Botão	Atividade	Sinais Luzes	Voz sintetizada
Prod1		X	X						
Prod2	X						X		X
Prod3	X					X			
Prod4		X			X			X	
Prod5	X		X	X	X				X
Prod6		X					X	X	
Prod7	X		X	X			X		
Prod8		X			X			X	

Figura 1.2: Tabela de Produtos

Capítulo 2

Arquitetura

2.1 Noções sobre a Arquitetura do Sistema

A arquitetura do sistema foi construido utilizando o *middleware Bezirk* fornecido. Todos os nossos sensores e atuadores encontram-se conectados ao *middleware* permitindo um maior *decoupling* entre componentes.

Todos os sensores publicam os seus valores no *middleware Bezirk*, sendo que no nosso projeto alguns desses valores são gerados aleatoriamente para simular sensores verdadeiros. Cada atuador recebe respetivamente de cada sensor o evento e enviam para o output uma mensagem indicando a ação realizada. Para o output é obrigatoriamente utilizado o ecrã, sendo que pode ser complementado com luzes ou voz sintetizada. Para simplificação escolhemos uma persistência com objetos serializáveis dos alertas criados pelo utilizador e uma abordagem Gson para a lista de contactos que é estática.

A figura seguinte representa a arquitetura bem como o tipo de eventos que são publicados e subscritos. Os eventos publicados antecedem um ponto de exclamação, os eventos subscritos antecedem de um ponto de interrogação.

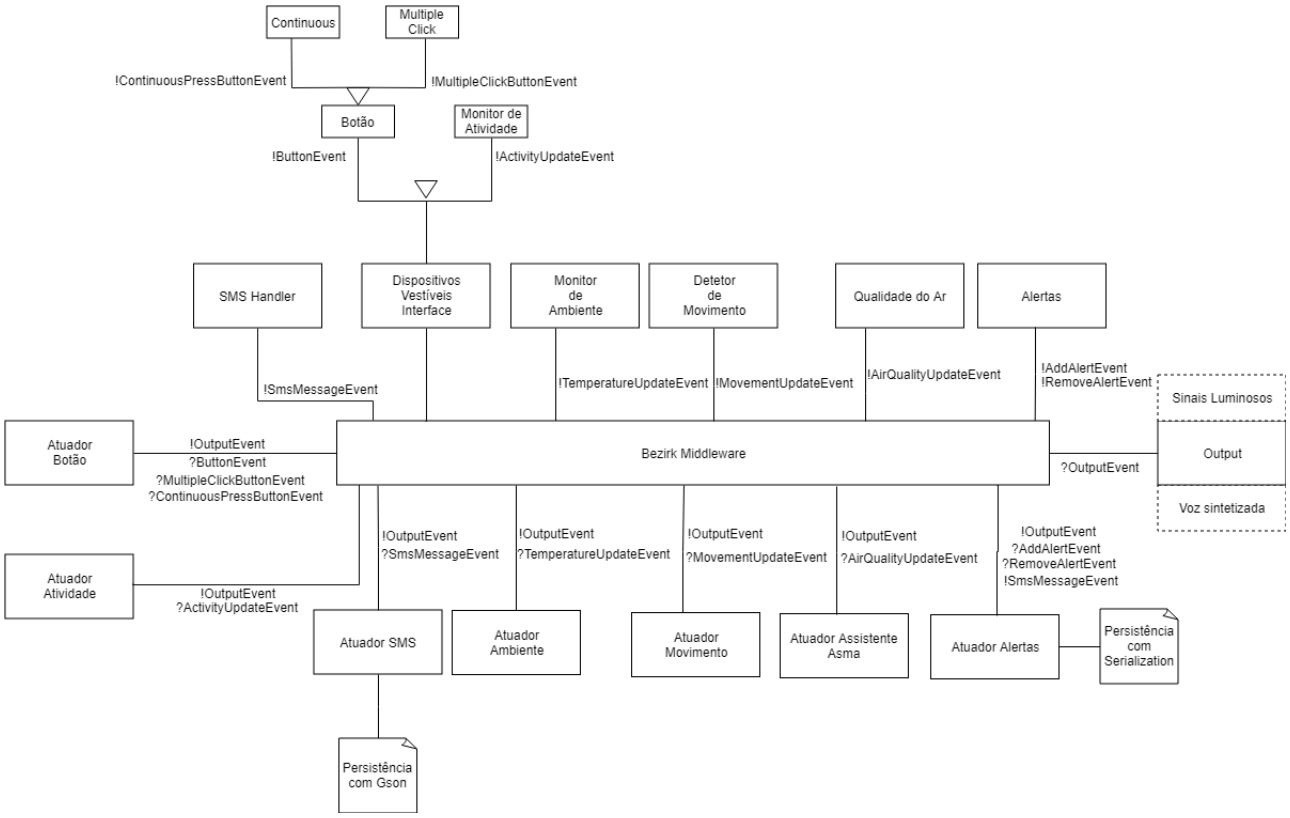

Figura 2.1: Arquitetura do Sistema

2.2 Vista de Módulos

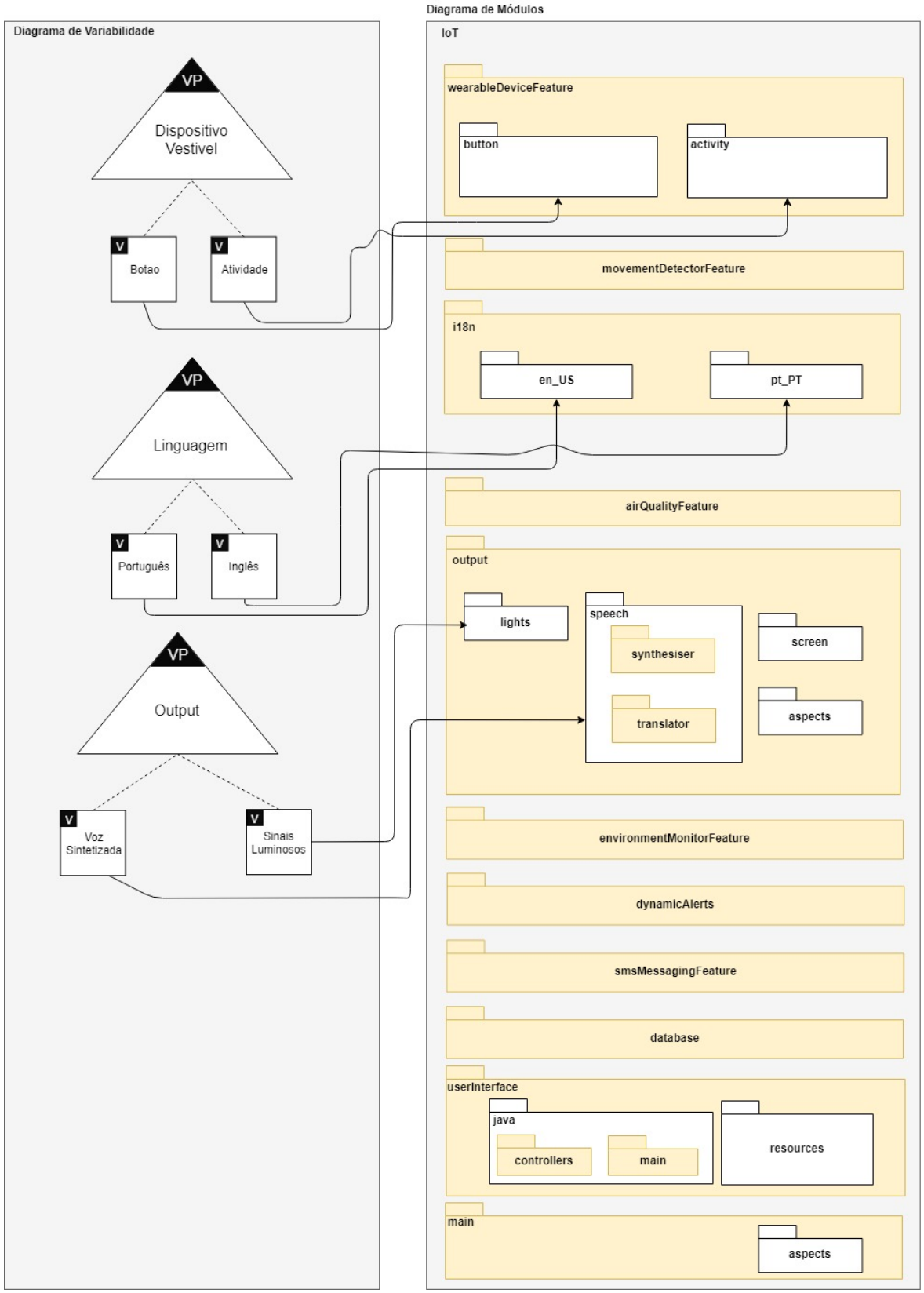

Figura 2.2: Vista de Módulos do Sistema

Na figura 2.2 apresentamos uma vista de módulos correlacionada com a variabilidade do sistema.

A funcionalidade de botão e atividade do dispositivo vestível encontram-se, respetivamente, nos módulos *button* e *activity* e ambos no módulo correspondente aos dispositivos vestíveis, *wearableDeviceFeature*.

Existe também um módulo para o detetor de movimento, *movementDetectorFeature* onde se encontram implementados os *Zirks* e eventos relacionados com a deteção de movimento para além de um *motion sensor* implementado usando a câmara do dispositivo.

A deteção de qualidade do ar, *airQualityFeature*, e o monitor de ambiente, *environmentMonitorFeature* representam as *features* que contêm os sensores relativos à recolha de informação do meio ambiente e que interatuam de acordo com a informação recolhida.

Os alertas que são definidos pelo utilizador encontram-se no pacote *dynamicAlerts*, onde o atuador de alertas envia eventos para o atuador de mensagens SMS. Este serviço de sms's situa-se no pacote *smsMessagingFeature*.

A base de dados que contém os contactos encontra-se no pacote *database* onde realizam a persistência de uma lista pré-definida dos contactos.

No pacote *userInterface* temos a interface com a qual o utilizador recebe mensagens e onde a qual pode adicionar alertas. Este é composto pelos sub-pacotes java, que contem os controladores e o Startup para arrancar com a interface.

Existe ainda um pacote *main* para o Main que irá correr todo o sistema e os aspetos correspondentes.

Existe um pacote i18n, constituído por sub-pacotes en_US e pt_PT, para a internacionalização dos produtos.

Por fim temos um pacote output para os vários tipos de outputs que o sistema tem, as luzes no pacote lights, a voz sintetizada no speech, o ecrã e por fim um pacote com todos os aspetos relativos a cada um.

Relativamente à variabilidade, o sistema apresenta vários pontos de variabilidade representados também nesta figura. Um dispositivo vestível pode ter um botão ou um monitor de atividade, a linguagem pode ser inglesa ou portuguesa e o output pode ser feito através de voz sintetizada ou sinais luminosos. Esta variabilidade é mantida pelos ficheiros *ajproperties* que criámos para cada produto deixando apenas os pacotes necessários ao produto evitando *dead code* e a chamada das *features* é feita através de aspetos que na inicialização controlam as funcionalidades que devem de ser criadas.

2.3 Vista de Componentes e Conectores

2.3.1 Vista C&C do Monitor de Movimento

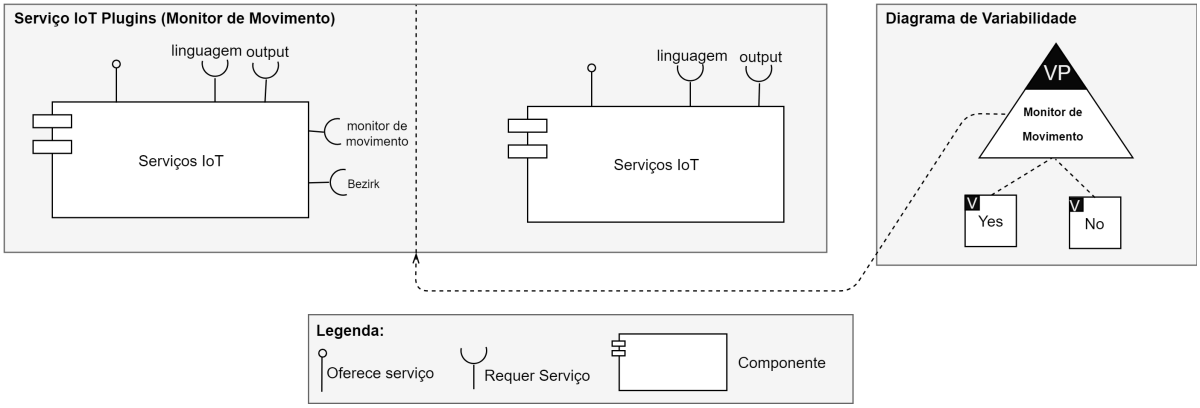

Figura 2.3: Vista de Componentes e Conectores do Monitor de Movimento

O ponto de variabilidade do monitor de movimento corresponde em ter ou não esta funcionalidade. Caso o produto apresente esta funcionalidade, então os nossos Serviços IoT vão necessitar de um componente do monitor de movimento, do middleware *Bezirk* para troca de eventos e também da instancia i18n que contém a linguagem na qual os eventos serão representados. Utilizando a exclusão dos ficheiros *ajproperties* foi possível adicionar ou remover esta funcionalidade sem apresentar *dead code*.

2.3.2 Vista C&C do Monitor de Ambiente

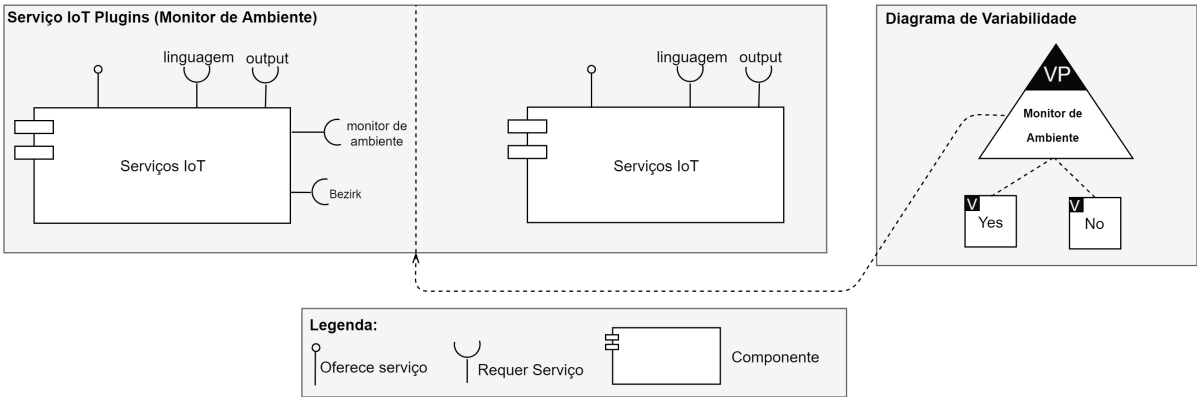

Figura 2.4: Vista de Componentes e Conectores do Monitor de Ambiente

O ponto de variabilidade do monitor de ambiente corresponde em ter ou não esta funcionalidade. Tal como o Monitor de Movimento, caso o produto apresente esta funcionalidade, então os nossos Serviços IoT vão necessitar de um componente do monitor de ambiente com sensores que recolhem dados, do middleware *Bezirk* para troca de eventos e também da instancia i18n que contém a linguagem na qual os eventos serão representados. Utilizando a exclusão dos ficheiros *ajproperties* foi possível adicionar ou remover esta funcionalidade sem apresentar *dead code*.

2.3.3 Vista C&C do Output

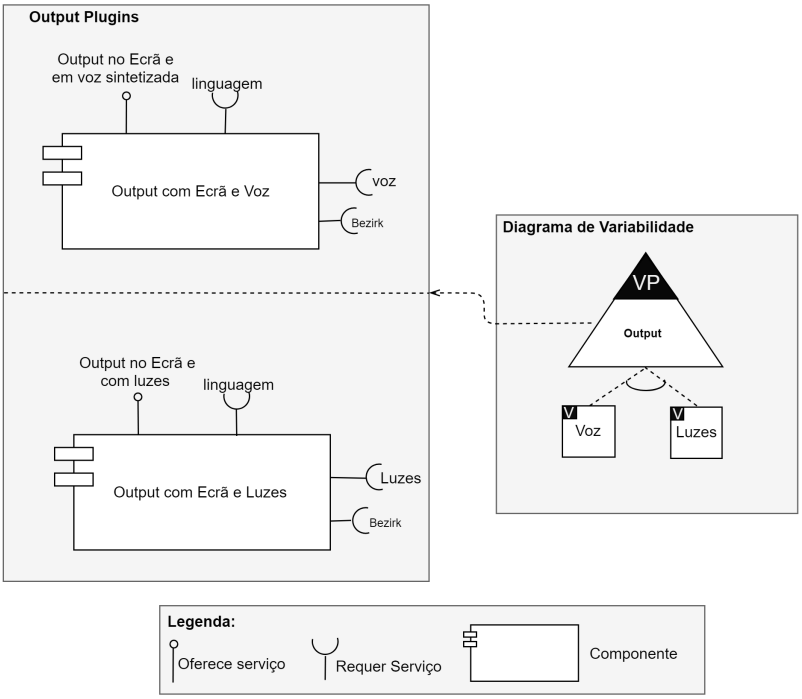

Figura 2.5: Vista de Componentes e Conectores do Output

O ponto de variabilidade do Output traduz-se em poder ter opcionalmente ou voz sintetizada ou sinais de luzes.

A combinação com o ecrã irá traduzir-se em dois componentes diferentes, sendo o primeiro componente de ecrã e voz sintetizada que requer uma linguagem, uma API para realizar a sintetização da voz e o *Bezirk* para receber as mensagens vistas de outros componentes e reproduzindo a mensagem no ecrã e através da voz.

O componente com ecrã e luzes requer também uma linguagem definida no i18n para apresentar na consola a mensagem apropriada, e também os uma API que permita realizar sinais de luzes. É também necessário *Bezirk* para a subscrição de eventos de output.

2.3.4 Vista C&C do Monitor de Qualidade do Ar

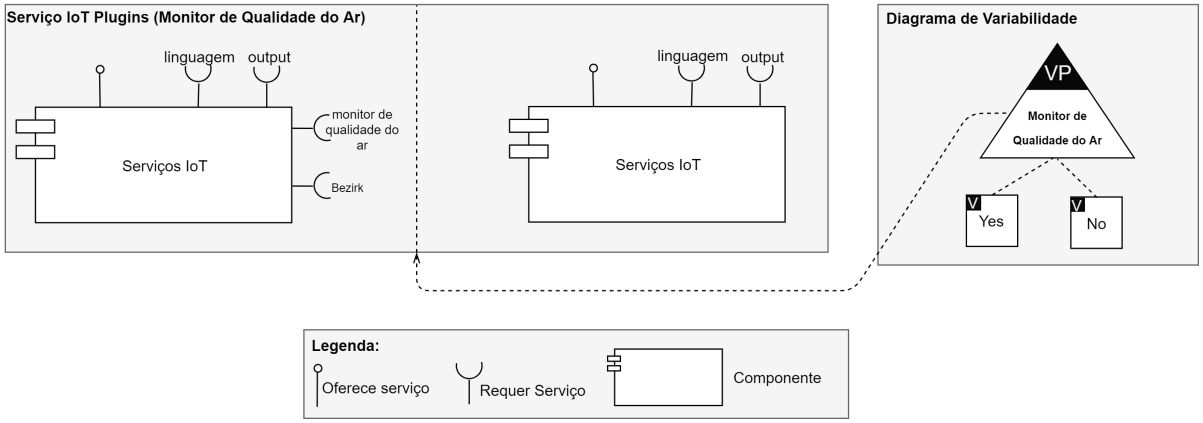

Figura 2.6: Vista de Componentes e Conectores do Monitor de Qualidade do Ar

O ponto de variabilidade do monitor da qualidade de ar corresponde em apresentar ou não esta funcionalidade. Caso o produto apresente esta funcionalidade, então os nossos Serviços IoT vão necessitar de um componente do monitor de qualidade de ar com sensores que recolhem dados sobre humidade, pó e polen. Também necessitará do middleware *Bezirk* para troca de eventos e também da instância i18n que contém a linguagem na qual os eventos serão representados. Utilizando a exclusão dos ficheiros *ajproperties* foi possível adicionar ou remover esta funcionalidade sem apresentar *dead code*.

2.3.5 Vista C&C do Dispositivo Vestível

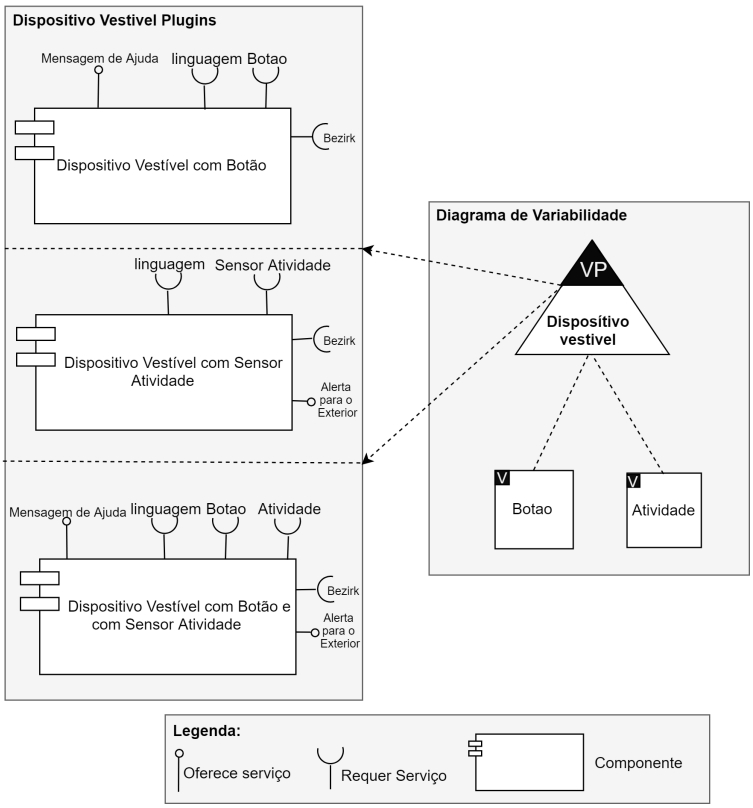

Figura 2.7: Vista de Componentes e Conectores do Dispositivo Vestível

O ponto de variabilidade do dispositivo vestível prende-se em ter botão ou monitor de ambiente ou ambos ou até nenhum.

Para o componente de dispositivo com botão é requerido o *Bezirk* para publicação e subscrição de eventos de botão premido continuadamente e premido várias vezes e o serviço do Botão que contém toda a lógica e devolve um alerta no output desejado.

Para a funcionalidade do dispositivo vestível com monitor de atividade é requerido o *Bezirk* para troca de eventos, a linguagem definida no i18n para definir o modo de representação do output e o sensor de atividade que lança alertas de ajuda para o exterior no output desejado e em sms.

Utilizando a exclusão dos ficheiros *ajproperties* foi possível adicionar ou remover um dispositivo vestível com botão ou monitor de atividade ou com ambos.

2.3.6 Vista C&C da Linguagem

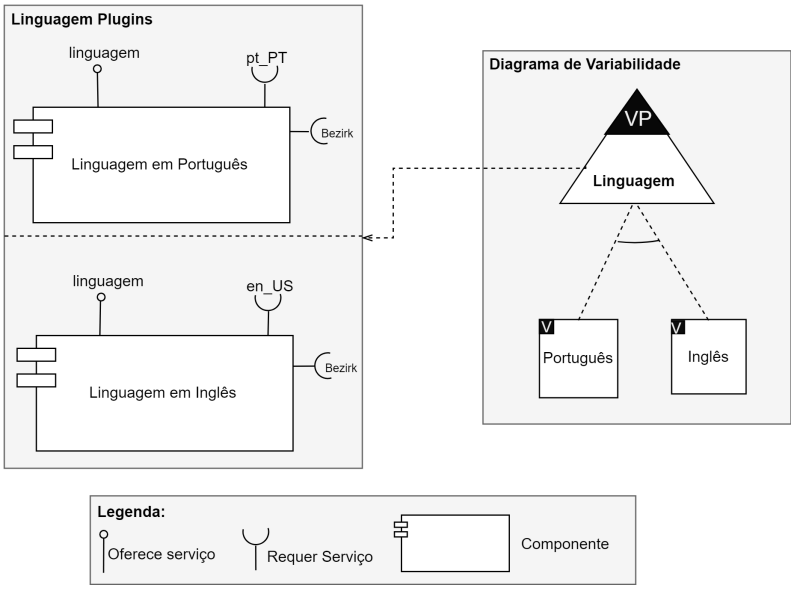

Figura 2.8: Vista de Componentes e Conectores da Linguagem

O ponto de variabilidade da linguagem prende-se em ser em português ou inglês e ser uma funcionalidade obrigatória do sistema.

Em cada caso é necessário o *Bezirk* para definir o output em que essa linguagem irá ser utilizada e o aspeto da linguagem em consideração que se ira encarregar de criar uma nova instância do i18n. Essa instância será utilizada por todos os serviços IoT para apresentarem o seu output na cuja linguagem.

Utilizando a exclusão dos ficheiros *ajproperties* foi possível escolher uma das linguagens para a instância do sistema.

Capítulo 3

Decisões de Implementação

3.1 Decisões tomadas na Implementação

Durante a implementação do projeto foram tomadas várias decisões para realizar o enquadramento das funcionalidades pretendidas.

Realizou-se a concretização de cada *feature* no seu respetivo módulo. Cada módulo contém na sua generalidade um sensor que recolhe informações e publica-os para o *Bezirk*, um atuador que recebe os eventos publicados pelo sensor e realiza o seu tratamento podendo enviar ou não alertas para o utilizador e finalmente eventos da *feature* que representam os eventos que são publicados pelo sensor.

Apresentamos no pacote *main* duas classes que permitem realizar determinadas operações sobre o sistema. Correndo a classe CreateProductsMain existe a possibilidade de criar um produto à escolha do utilizador. Após realizar a construção da configuração do produto criado, é possível correr o sistema através da classe Main existente também dentro deste pacote. Após indicar os valores para os sensores, uma interface gráfica aparecerá para receber os eventos dos sensores.

Os quatro produtos indicados anteriormente encontram-se também já concretizados, pelo que é apenas necessária realizar a construção da configuração e correr o Main fornecido.

Em seguida temos a descrição da implementação de cada uma das *funcionalidades*:

- **Detetor de Movimento:** O detetor de movimento foi implementado utilizando uma **Webcam Capture API**, que se liga à câmara do dispositivo e sempre que deteta movimento lança um evento para o *Bezirk*. Caso o dispositivo onde estejamos a correr o programa não suporte a API fornecida então é utilizado um *mock* que simula a deteção de movimento.

- **Mensagens de SMS:** As mensagens de SMS foram implementadas utilizando uma conta gratuita no **Twilio**. Infelizmente existe um limite de 300 SMS que podem ser enviados e só podem ser enviados SMS's para números de telemóvel que sejam registados na conta gratuita. Para prevenir o lançamento excessivo de SMS's, as linhas 25 e 26 do SMSHandler no pacote smsMessagingFeature foram comentadas. Descomentando estas linhas serão enviados SMS's dos alertas criados manualmente pelo utilizador para o número de telemóvel do Paulo Santos.

  Para além da implementação dos SMS's também existe um *mock* que finge que envia SMS's para uma lista de contactos já pré-definida.

- **Voz sintetizada:** A voz sintetizada foi concretizada utilizando uma **Google Speech API**. Esta API permite realizar a voz sintetizada em qualquer linguagem.

- **Luzes:** Foi utilizado um *mock* que imprime na consola uma mensagem a dizer que as luzes encontram-se a piscar.

- **Alertas:** Existe a possibilidade de um utilizador criar e apagar alertas personalizados. Estes alertas são criados pelo utilizador introduzindo as informações necessárias (designação, data de inicio, data de fim e intervalo). No acto da criação é associado um id ao alerta com o qual é possível realizar posteriormente a remoção do mesmo.

- **Qualidade do Ar:** Foi utilizado o *mock* que já vinha no projeto IoT que ocasionalmente envia um evento da qualidade do ar como a quantidade de pó, pólen e humidade e ao qual foi modificado para enviar esse evento para o Output e lhe foi retirado o main.

- **Detetor de Temperatura (Ambiente):** Encontra-se implementado utilizando um *mock* que envia aleatoriamente um valor da temperatura e cujo assistente responde em função de limites inferior e superior que são dados pelo utilizador.
- **Dispositivo Vestível com Botão:** Foi concretizado utilizando um *mock* que ocasionalmente escolhe um tipo de ação, botão pressionado continuamente ou pressionado múltiplas vezes.
- **Dispositivo Vestível com Atividade (Ritmo Cardíaco):** Encontra-se implementado utilizando um *mock* que envia um valor aleatório do ritmo cardíaco e ao qual o atuador responde em função de limites definidos pelo utilizador para os batimentos cardíacos.

3.2 Decisões relativas à visualização do Output

Relativo ao output começámos por mostrar todo o output na consola. Isto era feito através da classe OutputZirk que recebia OutputEvents e imprimia a mensagem correspondente. Se o produto tiver voz sintetizada ou recorrer a sinais luminosos é ao imprimir na consola que os aspetos que definimos correm.

Decidimos posteriormente criar uma interface gráfica. Para isto recorremos ao JavaFX e aplicámos os conhecimentos de CSS. O output na interface é apresentado numa janela que mostra todas as mensagens que recebe. Existem também dois botões que permitem criar ou apagar alertas.