Ciências ULisboa | Informática

Design de Software

Mestrados em Informática e Engenharia Informática

Trabalho Prático 2

2022/2023

O objetivo deste trabalho é fundamentalmente propiciar aos alunos uma experiência:

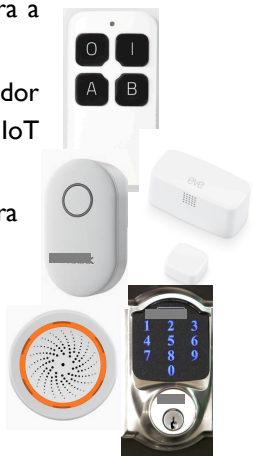
- na conceção e implementação duma linha de produtos de software e também na utilização de diferentes mecanismos de gestão da variabilidade.

O trabalho deve ser realizado em grupo de 2 a 4 elementos, é cotado para 20 valores e vale 20% da nota final da disciplina. A entrega é feita através da página da disciplina, **até 21 de Dezembro**. Após a entrega cada grupo deve marcar uma data com o docente para fazer a demonstração e discussão do trabalho realizado.

Problema. Suponha que uma empresa pretende explorar a cada vez maior quantidade de aparelhos IoT disponíveis no mercado e apostar no desenvolvimento de sistemas que ajudem pessoas com diferentes limitações a ultrapassarem dificuldades comuns no seu dia-a-dia e, ao mesmo tempo, deem às suas famílias ou cuidadores alguma tranquilidade relativamente ao seu bem-estar.

Porque diferentes funcionalidades exigem a utilização de diferentes tipos de aparelhos IoT, a opção pelo desenvolvimento de uma linha de produtos foi considerada a melhor forma de ter, a baixo custo, uma oferta de produtos, em termos de funcionalidade e preço, alinhada com as necessidades de diferentes tipos de clientes, com diferentes necessidades.

Requisitos. Após um estudo de mercado, foram definidos os seguintes requisitos para a linha de produtos a desenvolver.



- Cada produto da linha é uma aplicação IoT que é lançada a partir de um computador pessoal com ligação à internet e a uma rede local *wifi*, onde serão ligados os aparelhos IoT e *gateways* necessários ao funcionamento da aplicação.
- Os aparelhos IoT necessários ao funcionamento da aplicação variam de produto para produto, dentro das seguintes possibilidades:
  - Campainha de porta inteligente sem fios, que deteta quando é pressionada
  - Sensor de contacto de porta sem fios, que deteta quando a porta é aberta e fechada
  - Fechadura inteligente com teclado numérico sem fios, que permite vários códigos e acesso e aceita comandos de abertura de porta
  - Comando inteligente de 4 botões sem fios
  - Lâmpadas inteligentes, que aceitam comandos para se acenderem e apagarem
  - Sirene inteligente sem fios, com pelo menos 2 sons diferentes, que pode ser ativada com um som e desativada; opcionalmente a sirene pode ter luzes *led* ativadas ao mesmo tempo que o som

- Há produtos da linha específicos para o mercado de língua portuguesa e inglesa.
- Há produtos da linha específicos para pessoas com problemas de audição. Estes produtos têm necessariamente lâmpadas ou uma sirene inteligente com leds.
- Todos os produtos da linha enviam informação em tempo real para o exterior, a um contacto definido para o efeito. Existem produtos em que esta informação segue na forma de mensagens *sms* e outros em que segue através de mensagens *whatsapp*.
- Os produtos da linha fazem diferentes tipos de avisos para o interior da casa. Todos os avisos são mostrados no ecrã do computador. Nos produtos da linha que não são específicos para pessoas com problemas de audição é usada também voz sintetizada. Nos produtos da linha específicos para pessoas com problemas de audição, são usados sinais de luzes para sinalizar que existe um aviso novo.
- Todos os produtos da linha têm uma funcionalidade que permite criar e apagar *lembretes*. Ao definir um lembrete, o utilizador tem de indicar o título, o período em que o lembrete vai estar ativo e com que periodicidade se repete, se for esse o caso. O sistema faz um aviso para cada lembrete ativo. Uma mensagem com o lembrete é também enviada para o exterior.

  Exemplo de definição de lembrete num produto de língua inglesa:

  - *Title Take blue pill Starts 2022-12-02\|08:00 Ends 2022-12-20\|20:00 Repeats Every 8h*

  Exemplo de mensagem com o lembrete num produto de língua inglesa:

  - *MSG:Reminder: Take blue pill at 8:00 on Friday, Dec 2.*
- Os produtos com campainha têm a funcionalidade *tocar à campainha*: num sistema com sirene, faz soar a sirene sempre que a campainha é pressionada; caso contrário, o sistema faz um aviso.
- Os produtos que têm sensor de contacto de porta têm a funcionalidade *deteção porta aberta:* permitem definir quanto tempo a porta aberta precisa de estar aberta para ser despoletado um aviso e fazem esses avisos.
- Os produtos que têm campainha e sensor de contacto de porta têm a funcionalidade *deteção visitante:* permitem definir o período do dia em que a abertura de porta após o toque da campainha deve despoletar o envio de uma mensagem para o exterior.
- Todos os produtos com um comando inteligente têm a funcionalidade *pedido de ajuda*: enviam uma mensagem para o exterior quando o botão 1 é pressionado. A pessoa é também avisada que está a emitir um pedido de ajuda.
- Nos produtos que têm um comando inteligente e lâmpadas inteligentes o botão 2 serve para apagar as luzes; nos produtos com sirene o botão 3 serve para ativar a sirene; nos produtos com fechadura inteligente, o botão 4 serve para abrir a porta.

Foi também já decidido que:

- De forma a facilitar o desenvolvimento das aplicações, nomeadamente a interoperabilidade entre diferentes dispositivos, será usado o *middleware Berzik*, fornecido no material de apoio.
- A aplicação deve ficar dependente apenas de software de acesso livre.
- Todos os dados recolhidos pelos aparelhos de IoT, assim como outros dados recolhidos pelos produtos da família, são para uso exclusivo da aplicação, guardados localmente e não partilhados.

Tarefas

Pretende-se que conceba e implemente uma primeira versão do software desta linha de produtos. Nesta versão não vai ser preciso ainda lidar com a incorporação de dispositivos de hardware concretos. Uma vez que foi tomada a decisão de usar o *Berzik,* é preciso apenas definir as interfaces apropriados dos elementos que encapsulam a comunicação com os dispositivos de hardware que irão estar envolvidos e fornecer implementações *mock* dos mesmos (preparados para correr em processos independentes). Note que no caso dos *mocks* de dispositivos com sensores pode usar uma leitura do standard input para simular uma medição do sensor.

A interface dos produtos pode ser textual e muito simples. Porém é muito importante que a interface de cada produto da linha esteja de acordo com as funcionalidades que existem nesse produto e que sejam usados mecanismos apropriados para a gestão da variabilidade que também existe a este nível. Todo o código da interface dos produtos deve estar num módulo que não tem mais nenhuma espécie de responsabilidade e que pode ser substituído por um módulo que implemente uma interface gráfica.

Algumas funcionalidades exigem persistência de dados. Esta persistência pode ser concretizada de forma muito simples, por exemplo recorrendo a ficheiros de texto, mas é importante que seja concretizada.

Mais especificamente devem realizar as seguintes tarefas:

1. Conceber um modelo de características (*feature model*) para a linha de produtos que permita à empresa ter uma oferta de produtos tão diversificada quanto possível. Todos os requisitos descritos na página anterior devem ficar cobertos. Não devem esquecer de explicitar as restrições aplicáveis à combinação das funcionalidades constantes nesse modelo.
2. Apresentar um quadro com alguns dos produtos da linha e as suas características, incluindo os quatro produtos que decidirem implementar (ver ponto 4. abaixo).
3. Conceber uma solução de desenho para a linha de produtos, que
   - utilize o *Bezirk* para tratar da comunicação com os dispositivos de *hardware*
   - utilize *aspetos* para gerir a variabilidade, em pelo menos alguns pontos de variação
   - utilize *injeção de dependências* para gerir a variabilidade em pelo menos um ponto de variação

   e documente a arquitetura da sua solução, nomeadamente através de:
   - vistas de módulos que, além do que é comum neste tipo de vistas, explicitem a correspondência entre os módulos e as *features* do modelo
   - vistas de componentes e conectores que forneçam o contexto, explicitem os pontos de variabilidade e documentem os mecanismos usados para concretizar a variabilidade suportada
   - vista de *deployment*

   Antes de arrancar com a implementação devem pedir feedback relativamente à solução que conceberam de forma a evitarem desperdiçar o vosso tempo a escrever código que não implementa uma solução para o problema proposto e que, portanto, não terá qualquer cotação. Além das duas aulas dedicadas ao apoio a este trabalho, podem usar o horário de atendimento semanal.
4. Implementar um protótipo da linha de produtos que permita demonstrar **quatro produtos** diferentes da linha de produtos. Estes quatro produtos devem incluir:
   - um produto para o mercado português
   - um produto para o mercado inglês
   - um produto específico para pessoas com problemas de audição
   - um produto específico para pessoas sem problemas de audição
   - um produto que tem um comando inteligente
   - um produto que tem campainha de porta e detetor de abertura de porta

   A implementação deve ser fornecida na forma de um projeto contendo:
   - O código, bibliotecas, e outros recursos necessários para algum dos produtos da linha.
   - Algo que permita facilmente construir e executar cada um dos 4 produtos escolhidos. Idealmente, a construção desses produtos deve incluir apenas os módulos de código estritamente necessários para concretizar as suas características (ou seja, não deve ter *dead code*). Pode, por exemplo, usar ficheiros de configuração do *build* ou usar os ficheiros de configuração do FeatureIDE.
   - Um README com a descrição do projeto

Informação Adicional

- O middleware *Berzik* foi desenvolvido pela *Bosch* e na altura disponibilizado para utilização livre. Entretanto a informação sobre o projeto em http://developer.bezirk.com/ e o código em https://github.com/Bezirk-Bosch/AdapterZirks deixou de estar disponível, pelo que a informação necessária para se familiarizarem com o *Berzik* é fornecida no material de apoio ao trabalho. Estão ainda disponíveis duas apresentações sobre o Bezirk no SATURN 2016 aqui:
    - https://www.youtube.com/watch?v=UWd35FUGFcM&list=PLgJhofPm90nnIY3mY4Y0caGQNwjYwU2k5&index=2
    - https://www.youtube.com/watch?v=aoCjVh_z2y8&list=PLgJhofPm90nnIY3mY4Y0caGQNwjYwU2k5&index=1
- Os projetos exemplo do *Berzik* usam o *gradle*. Como o uso simultâneo do *gradle* e *AspectJ* no Eclipse é potencialmente problemático, é fornecido um projeto exemplo que não precisa do *gradle* e que usa o *AspectJ*. O projeto é uma adaptação do exemplo descrito no tutorial 2 do *Berkiz* (incluído no material de apoio) e envolve:
    - um *mock* de um *Air Quality Sensor Zirk* que publica dados sobre a qualidade do ar — humidade, nível de poeiras e nível de pólens— que num sistema real serão recebidos de um aparelho IoT com a capacidade de medir e transmitir esses dados;
    - um *Asthma Assistant Zirk* que subscreve os dados de qualidade do ar e usa-os para dar instruções ao utilizador de como proceder (por exemplo, ligar o desumidificador);
    - um conjunto de classes, aspectos e outros recursos que permitem construir uma variante do sistema em que as instruções ao utilizador são dadas em português e outra variante em que são dadas em inglês

  Para correr um dos sistemas, depois de escolher a *build configuration* apropriada, deve proceder como está descrito no tutorial 2. Porque a comunicação suportada pelo *Berzik* é realizada através de uma rede *wifi*, certifique-se que está ligado a uma.

  Importante: O exemplo serve só para ilustrar a utilização do *Berzik*. Nos produtos da linha que têm de implementar vão existir vários sensores e a interpretação dos dados recolhidos por estes sensores vai poder variar de produto para produto. Além disso, a parte da interface dos produtos deve estar separada de tudo o resto.

  Utilize o código fonte do *berzik-middleware-api* para dentro do Eclipse ter acesso ao seu *javadoc*.
- Para demonstrar um produto que exija dispositivos de hardware tem apenas de programar *mocks* apropriados (tal como foi feito no exemplo fornecido). Para ficar com uma ideia de como, recorrendo a adaptadores, é suportada no *Berzik* a integração de dispositivos de hardware, por exemplo com as lâmpadas inteligentes da Philips, pode olhar para o projeto *AdapterZiker*/*PhilipisHue* fornecido no material de apoio. Pode aproveitar ainda para olhar para os eventos que estas lâmpadas publicam e subescrevem de forma a que lhe sirvam de inspiração para os tipos de eventos que vai considerar na sua implementação.
- Para demonstrar um produto que exija síntese de fala pode usar por exemplo o framework *FreeTTS* http://freetts.sourceforge.net/docs/index.php ou por exemplo esta API Java para usar os serviços da Google https://github.com/goxr3plus/java-google-speech-api
- Para demonstrar um produto que exija envio de mensagens de *sms* ou *whatsapp* não precisa de concretizar o envio efetivo das mensagens, mas para experimentar pode por exemplo usar o *Twilio* https://www.twilio.com/docs/sms/quickstart/java