Aplicação Móvel Consciente do Contexto para Consulta das Ementas das Cantinas das Escolas do IPP

Pedro Emanuel
MEI-ISEP
1131485@isep.ipp.pt

João Freitas
MEI-ISEP
1160907@isep.ipp.pt

RESUMO

A alimentação tomada pelos estudantes universitários é um aspeto bastante importante que deve ter um cuidado especial por estes. A falta de informação sobre locais de restauração que disponibilizem uma boa alimentação pode levar a que os estudantes optem por uma má alimentação. O presente artigo descreve a conceção de uma aplicação móvel Android que permite aos estudantes e cidadãos da área do Grande Porto consultar as ementas disponíveis nas cantinas das escolas do IPP, abordando a contextualização, análise e design desta.

Palavras-Chave

Alimentação; Aplicações Móveis; Android; Estudantes Universitários; Ementas; IPP

1. CONTEXTO

A alimentação que é tomada pelos humanos é um aspeto bastante importante que deve ter um cuidado especial. Esta tem um grande impacto no decorrer do dia-a-dia do humano, sendo que o mau cuidado da alimentação contribui para o aumento do stresse, cansaço e capacidade de trabalho, a curto prazo, e pode levar ao risco do desenvolvimento de doenças crónicas tais como a obesidade, colesterol, hipertensão e depressão [1], [2]. O tema da alimentação torna-se ainda mais importante nos estudantes universitários pois pode afetar bastante negativamente o progresso que estes têm durante o seu percurso académico, podendo levar até à desistência deste devido ao stress, ansiedade e depressão. O não conhecimento de locais de restauração, onde se possa almoçar, lanchar e jantar, pode levar a que os estudantes se dirijam diariamente a locais que providenciem uma má alimentação, tais como cadeias de fast-food [3].

De acordo com um estudo efetuado em outubro de 2019 sobre o impacto das aplicações móveis no mundo universitário [4], foi concluído que o uso destas pelos estudantes reduzia o tempo perdido na decisão do que este se podia alimentar. Foi também notado que os estudantes da Universidade do Porto, não tem o acesso à informação da ementa das diversas cantinas das universidades automatizado através da aplicação móvel UPorto. Relativamente às escolas do Instituto Politécnico do Porto (IPP), de momento também não existe uma aplicação funcional que permita acesso a esta informação [5].

Tendo em conta o impacto da alimentação nos estudantes universitários e a inexistência de automatismos ao acesso da informação das ementas das cantinas das escolas do IPP, foi concebida o *ipp-ementa*, uma aplicação móvel Android que irá permitir a consulta das ementas das cantinas das escolas do IPP. Sendo uma aplicação de uso aberto, não só os estudantes das escolas do IPP irão poder consultar as ementas das suas escolas, como também outros estudantes e cidadãos da área do Grande Porto irão poder fazer a consulta. De modo a melhorar a experiência do estudante, a aplicação móvel é também consciente do contexto, permitindo assim que no decorrer do dia-a-dia do estudante este receba diversas notificações informativas sobre cantinas disponíveis na sua proximidade e também sobre ementas das quais o estudante tem preferência.

2. ANÁLISE

De modo a compreender e identificar os conceitos de negócio do ipp-ementa, foi elaborado um conjunto de artefactos que ilustram o domínio da aplicação. As subseções seguintes expõem estes artefactos, abordando os requisitos funcionais, preocupações e atributos de qualidade da aplicação, como também os respetivos pressupostos e restrições que tem de ser cumpridas. Apesar de este documento estar escrito em Português, os artefactos estão em Inglês visto que é o idioma de preferência por parte dos desenvolvedores.

2.1 Requisitos Funcionais

De modo a expor todas as funcionalidades que a aplicação permite realizar, recorreu-se a um diagrama de casos de uso, na qual este representa todas as funcionalidades na forma de caso de uso, das quais estão atribuídas ao ator que as pode realizar. A figura abaixo ilustra este diagrama.

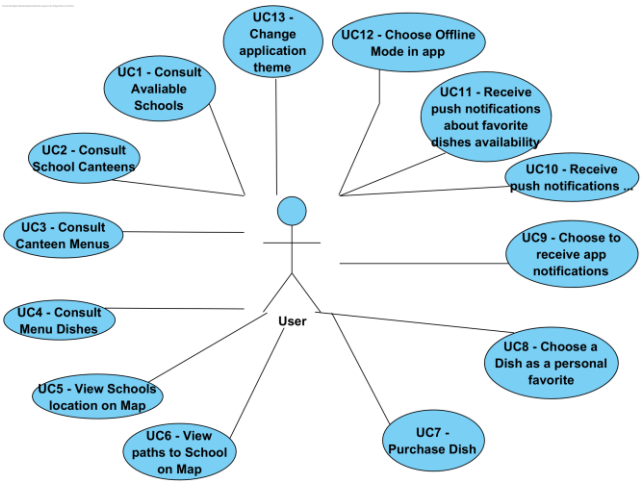

Figura 1 - Funcionalidades do ipp-ementa representadas através de um diagrama de casos de uso em UML

Através do diagrama apresentado, é possível reter que existem quatro casos de uso de consulta de informação (UC-1, UC-2, UC-3 e UC-4), permitindo assim a listagem das escolas disponíveis, cantinas, ementas e respetivos pratos na aplicação. Estas são as funcionalidades *core* da aplicação pois irão permitir ao utilizador

consultar a informação das ementas das cantinas das escolas do IPP.

De modo a melhorar a experiência do utilizador foram concebidos quatro grupos de funcionalidades adicionais, sendo estes identificados por *Mapas*, *Customizações*, Notificações e Compras. Nas funcionalidades do grupo *mapas* (UC-5 e UC-6), o utilizador irá poder visualizar a localização de uma cantina num mapa e também a rota desde a localização do utilizador até à cantina. As funcionalidades do grupo *customizações* (UC-8, UC-9, UC-12 e UC-13), permitem que o utilizador altere o esquema de cores da aplicação, através de temas, usar a aplicação em modo offline através do descarregamento prévio dos dados, a escolha de um prato como favorito e também a permissão para receber notificações na aplicação. Já nas funcionalidades do grupo *notificações* (UC-10 e UC-11) permitem com que o utilizador receba *push-notifications* sobre cantinas que se encontram nas proximidades do utilizador, recorrendo ao uso de estratégias de *geofencing* e também sobre a disponibilidade dos pratos que o utilizador selecionou como favorito. Por fim nas funcionalidades do grupo *compras* (UC-7), a aplicação irá substituir o uso de objetos monetários (e.g. nota, moeda) na compra de um prato, permitindo assim que o utilizador realize o ato de compra através da aplicação.

Todas as funcionalidades requerem o uso da Internet, pelo menos na sua primeira utilização.

2.2 Atributos de Qualidade

Apesar de não pertencer ao conjunto de funcionalidades das quais o utilizador pode interagir, é necessário na mesma definir um conjunto de atributos de qualidade de modo a complementar e melhorar de forma geral a aplicação. A tabela seguinte define os atributos de qualidade estipulados para o ipp-ementa.

Tabela 1 - Atributos de Qualidade do ipp-ementa

ID	Atributo Qualidade	Cenário
QA-1	Eficiência	O uso diário da aplicação não deve consumir mais que 10% de utilização de bateria do dispositivo móvel
QA-2	Localização	A aplicação deve providenciar feedback nos idiomas Português e Inglês (geral)
QA-3	Portabilidade	Deve ser possível instalar a aplicação pelo menos em 95% dos dispositivos Android existentes
QA-4	Auditabilidade	Todos os erros e exceções que ocorrem na aplicação devem ser auditados para análises futuras

2.3 Preocupações e Restrições

Tendo os requisitos funcionais e atributos de qualidade definidos é necessário reforçar estes através de um conjunto de preocupações e restrições. Entende-se como preocupação algo que afete a concretização do estado final da aplicação, mas que não seja obrigatório. Já nas restrições estas também afetam a concretização do estado final da aplicação, mas são de carater obrigatório. As tabelas 2 e 3 representam estas preocupações e restrições respetivamente.

Tabela 2 - Preocupações do ipp-ementa

ID	Preocupação
CRN-1	Suporte Multi-Idioma
CRN-2	*Logging* de erros e exceções

Tabela 3 - Restrições do ipp-ementa

ID	Restrição
CON-1	O uso diário da aplicação não deve consumir mais que 10% da utilização da bateria do dispositivo móvel
CON-2	Deve ser possível instalar a aplicação pelo menos em 95% dos dispositivos Android existentes
CON-3	A aplicação deve ser construída apenas com o uso de tecnologias open-source ou de sem custos

Como explicitado em ambas as tabelas, existe um conjunto de preocupações e restrições que complementam a obrigatoriedade dos atributos de qualidade (CRN-1, CRN-2, CON-1, CON-2 referenciam respetivamente QA-2, QA-4, QA-1 e QA-3), sendo que a restrição CON-3 não complementa nenhum atributo de qualidade. Esta tem apenas como responsabilidade restringir que a aplicação seja construída apenas com o uso de tecnologias open-source ou de sem custos, complementado diretamente todos os requisitos funcionais da aplicação. A imposição desta restrição deve-se ao facto que da mesma forma que a aplicação é distribuída de forma gratuita sem qualquer custo para o utilizador, esta também não pode trazer custos adicionais (para além do custo dos desenvolvedores) à organização do ipp-ementa.

2.4 Conceitos de Domínio

Uma vez definido os requisitos funcionais, atributos de qualidade, preocupações e restrições da aplicação é possível agora apurar todos os conceitos de domínio e respetivas relações.

Os conceitos de domínio principais que descrevem as funcionalidades *core* da aplicação descrevem-se por *School* (escola), *Canteen* (cantina), *Menu* (ementa) e *Dish* (prato), sendo que uma escola oferece pelo menos uma cantina, as cantinas podem disponibilizar ementas diárias e cada ementa é descrita por pelo menos um prato. Existe também o conceito de *User* que se entende como o utilizador da aplicação. Cada user pode selecionar

como favorito um conjunto de pratos e comprar estes mesmos, sendo que cada utilizador está numa localização num determinado momento. O utilizador pode também alterar entre dois temas na aplicação (ligth e dark theme), receber notificações da aplicação e escolher a utilização da aplicação em modo offline. Cada uma destas notificações da aplicação são representadas como *Push-Notifications*, sendo que de momento, estas apenas podem ser *Nearby Canteen Push Notification* e *Available Favorite Dish Push Notification*. As push-notifications relativas às cantinas que se encontram nas proximidades requerem o conhecimento da localização do utilizador e da cantina respetiva, estando assim relacionada com estes dois conceitos (*User Location* e *Canteen Location*). Já as available favorite dish push-notifications requerem o conhecimento dos pratos selecionados como favoritos pelo utilizador. Por fim o utilizador pode usar o mapa para visualizar a localização da cantina e as possíveis rotas desde a sua localização atual até à cantina.

3. DESIGN

Uma vez realizada a análise do domínio da aplicação, é necessário traduzir esta em conceitos de design. Da mesma forma que na análise, também para o design foram elaborados um conjunto de artefactos que ilustram este. As subseções seguintes expõem estes artefactos abordando os conceitos de design e respetivas decisões, como também os elementos arquiteturais que os expõem.

3.1 Conceitos de Design

De modo a satisfazer toda a análise da aplicação, foram escolhidos um conjunto de conceitos de design. A tabela 4 mapeia estes conceitos pelos requisitos funcionais, atributos de qualidade, preocupações e restrições.

Tabela 4 - Conceitos Design ipp-ementa

Conceito Design	Decisão
Suportar o *Android Kitkat* (Android API 19) como a mínima plataforma suportada pela aplicação	Segundo os dados dos desenvolvedores do Android, desde 7 de maio de 2019 apenas 3.8% dos dispositivos Android utilizam uma plataforma inferior à KitKat [6]. O suporte da plataforma Android KitKat irá permitir que o ipp-ementa seja suportado em 96.2% dos dispositivos, permitindo assim o cumprimento da restrição CON-2 e suporte para a concretização de todos os requisitos funcionais.
Utilizar o *Mapsforge* para visualização dos mapas e navegação das rotas	O Mapsforge [7] é uma biblioteca open-source e grátis que permite a renderização e escrita de *vector maps* para Android e Java. Esta funciona em modo offline, e consome os dados do *Open Street Map* [8] que estão pre-compilados num ficheiro. A utilização desta tecnologia cumpre com os requisitos CON-2 e CON-3 e suporta a concretização dos requisitos funcionais UC-5, UC-6 e UC-12 como também o atributo de qualidade QA-3
Utilizar o *Firebase Cloud Messaging* para emissão das push-notifications	O Firebase Cloud Messasing, ou FCM, é solução de entrega de mensagens em todas as plataformas que opera na cloud [9]. Este é de uso completamente gratuito [10], e irá suporta a concretização dos requisitos funcionais UC-10 e UC-11, atributo de qualidade QA-3 e cumprir com os requisitos CON-2 e CON-3.
Utiliza o *SQLite* para armazenamento dos dados da aplicação	O SQLite é uma base de dados relacional embebida [11], permitindo assim a persistência de dados num ficheiro armazenado na memória interna do dispositivo móvel. A utilização do SQLite irá permitir concretizar o requisito funcional UC-12, atributo de qualidade QA-3 e o cumprimento das restrições CON-2 e CON-3.
Adotar o padrão arquitetural MVP como a arquitetura da aplicação móvel	O padrão MVP (Model View Presenter) propõem a separação da responsabilidade do conhecimento dos dados, ao criar um intermediário (p*resenter*) entre a vista (*view*) e a informação dos dados (m*odel*). Este desacoplamento irá permitir uma melhor manutenção da *user interface*. A sua adoção irá suportar a concretização de todos os requisitos funcionais.
Utilizar *alternate resources* para suportar multi-idiomas	O Android providência um mecanismo de localização através do uso de alternate resources [12]. O uso deste mecanismo irá permitir a concretização do atributo de qualidade QA-2 e preocupação CRN-1.
Utilizar o *Room Persistence Library* para gestão da comunicação com a base de dados SQLite	O Room [13] providência uma camada de abstração sobre o SQLite, permitindo a comunicação com a base de dados através de uma *Fluent API*. O uso desta biblioteca irá permitir um desenvolvimento mais rápido da aplicação, removendo a responsabilidade na gestão da comunicação com a base de dados por parte da equipa de desenvolvimento. O Room irá suportar a

	concretização do requisito funcional UC-12, e também o cumprimento do requisito CON-1, visto que é solução testada, providenciada pelos desenvolvedores do Android.
Adotar o padrão de criação *Singleton* para centralizar o acesso ao contexto do objeto da base de dados	O Room providencia o acesso às funcionalidades da base de dados através de um objeto de nome *AppDatabase* que só deve criado uma vez por cada processo da aplicação [14]. A adoção deste padrão irá suportar o cumprimento do requisito CON-1.

3.2 Elementos Arquiteturais

De modo a satisfazer os conceitos de design escolhidos, foram *instanciados* os seguintes elementos arquiteturais:

IPEM – Componente que representa a aplicação móvel, produzindo uma interface gráfica de modo a que os utilizadores possam consumir esta. Este componente consome diversas interfaces externas de modo a permitir a concretização das funcionalidades, sendo estas:

- **IPED REST API**, produzida pelo componente **IPED** que permite o consumo das funcionalidades de lógica de negócio;
- **IPEPN Push Notifications Web API**, produzida pelo componente **IPEPN** que permite o pedido de início de emissão de push notifications para o dispositivo móvel;
- **IPEM Data**, produzida pelo componente **SQLite** que permite o armazenamento e consumo dos dados persistidos pela aplicação móvel;
- **NFC Purchase**, produzida pelo componente **School POS** que permite o registo do pagamento de um *dish* com o uso de NFC;
- **Bluetooth Purchase**, produzida pelo componente **School POS** que permite o registo do pagamento de um *dish* com o uso do Bluetooth;
- **GPS Data**, produzida pelo componente **GPS Sensor** que permite o consumo da localização do dispositivo;
- **Maps Data**, produzida pelo componente **Mobile Device Internal Storage** que permite o consumo dos dados do mapa a ser visualizado, presente na memória interna do dispositivo;
- **Firebase Push Notifications**, produzida pelo componente **Firebase Cloud Messaging** que permite o consumo em tempo real da emissão de push-notifications.

4. PRESSUPOSTOS

Uma vez realizada a análise e design do ipp-ementa, é possível definir um conjunto de pressupostos transversais na utilização da aplicação, impostos ao utilizador, sendo estes:

- A existência de um dispositivo móvel Android cuja versão da API seja superior ou igual a 19, bateria e Bluetooth ou NFC;
- Pelo menos 10MB disponíveis de memória interna;
- A existência de conetividade à Internet, pelo menos na primeira utilização de todas as funcionalidades;
- A compreensão da língua Portuguesa ou Inglesa.

5. IMPLEMENTAÇÃO

Uma vez realizada a análise e design da aplicação móvel, é necessário traduzir estes na implementação de modo a concretizar a aplicação móvel. As subseções seguintes elaboram a forma como os conceitos de design propostos foram abordados na implementação, utilização de padrões de software para redução do acoplamento, princípios de software seguidos, técnicas utilizadas para o controlo de processamento e memória utilizada e também sobre a forma como a *user-interface* foi estruturada.

5.1 Aplicação dos Conceitos de Design Propostos e Padrões de Software

Relativamente ao padrão arquitetural proposto a ser adotado (MVP), primeiramente foi estruturado o código da aplicação em três packages, model, view e presenter, que correspondem respetivamente às três entidades propostas pelo padrão. Em cada um destes packages foi criado um conjunto de packages identificados pelos vários conceitos de domínio, contendo nestes toda a implementação relativa ao domínio do respetivo conceito com as responsabilidades respetivas à entidade do padrão arquitetural. Como exemplo, a package *dish*, contida na package *view*, expõem toda a implementação relativa à user-interface e respetivas funcionalidades da qual o utilizador pode interagir relativamente ao conceito dos pratos de uma ementa (menu dishes).

Uma vez realizada a estruturação das packages, foi definido um conjunto de regras a respeitar de modo a não violar as regras do padrão MVP. Como mencionado na subseção 3.1, o padrão MVP tem como objetivo separar a camada de acesso aos dados da user interface. Para tal é concebido um intermediário, que atua como *mediador* entre estas camadas, falando com a camada de dados sempre que a user interface requisita estes, formatando os dados provenientes desta camada e por fim notifica a user interface que os dados estão prontos a ser apresentados. Dado a tecnologia e linguagem da aplicação móvel (Android e Java), para cada vista da aplicação, foi criado uma interface que expõem um conjunto de métodos, expondo todas as interações existentes nesta. Como exemplo, na interface *MenuDishesView* são expostos métodos tais como *showDishes* e *markDishAsFavorite* que permitem apresentar ao utilizador os pratos da ementa selecionada anteriormente e selecionar um prato como favorito. Esta interface é depois implementa pela atividade da vista correspondente. Ainda relativamente às vistas, foi também criado a interface *ErrorView* que expõem interações comuns a todas as vistas relativas a erros que podem ocorrer na interação das vistas (e.g. não existência de conexão à internet). Já no lado do intermediário (presenter), existe uma classe para cada view, que contém a lógica relativa à comunicação com a camada dos dados e respetiva formatação do resultado da comunicação. Cada presenter tem uma referência da view a ser gerida, cujo esta é recebido pelo construtor da classe. De modo a reforçar a consistência e o estado

da aplicação, foi também introduzida a variável booleana *isViewAvailableToInteract*, indicado se é possível interagir / utilizar a user interface. Esta toma o valor de falso caso a atividade que representa a vista esteja em pause (*onPause)* ou tenha sido "destruída" (onDestroy) pelo gestor das atividades do Android, permitindo assim cancelar a interação com a atividade uma vez necessária, eliminado o risco de ser lançada uma exceção. Um caso prático da utilização desta variável será por exemplo, a interação com alguma funcionalidade que necessite do uso da Internet, e que uma vez o acesso a esta se encontre lento, o utilizador decide cancelar esta e avança para uma nova atividade. Apesar de o utilizador estar agora numa nova atividade, a *thread* que processou a comunicação à Internet ainda não foi encerrada e uma vez terminada, irá tentar notificar o utilizador do fim desta. Ainda na camada intermediária, são também declarados um conjunto de classes que modelam os diversos dados a ser apresentados na user-interface, de modo a que esta não tenha necessite da preocupação de saber formatar os dados provenientes da camada de acesso aos dados.

Por fim na entidade respetiva à camada de acesso aos dados (model), foram aplicados os padrões *Repository* e *Factory* de modo a abstrair as várias possibilidades de acesso aos dados. Por exemplo, as cantinas de uma escola tanto podem ser consultadas através de o consumo da REST API produzida pelo componente IPED ou através base de dados SQLite3 que está embebida na aplicação. Para simplificar e unificar o acesso aos dados das cantinas, primeiramente foi concebido a interface *CanteensRepository* que expõem as interações existentes com o conceito de domínio *Canteen*. Seguidamente, esta interface é implementada pela classe *IPEDCanteensRepositoryImpl* e *RoomCanteensRepositoryImpl* que contém a lógica da comunicação com os componentes IPED e SQLite3 respetivamente. A interface *RepositoryFactory* unifica todos os repositórios existentes, permitindo a criação destes sem a necessidade de saber qual a implementação é que está a ser utilizada. Da mesma forma que no presenter, é também declarado no model um conjunto de classes que modela as respostas dos vários pedidos às APIs consumidas e de todos os dados existentes numa entidade de domínio (e.g. classe *School* declara o nome, acrónimo e lista de cantinas fornecida por uma escola). O output de cada método dos repositórios (se existente) é sempre uma instância de uma entidade de domínio. Para efetuar a serialização e deserialização dos dados provenientes das APIs consumidas, recorreu-se à utilização da biblioteca *Gson*, desenvolvida e mantida pela Google, que permite de forma ágil a tradução de strings em instancias de classes e vice-versa.

Também na package model foi declarado os respetivos serviços para o controlo da receção das mensagens provenientes do Firebase caso a aplicação esteja em modo *foreground* (*IPPEmentaFirebaseMessagingService*), como também o *BroadcastReceiver* que controla as transições dos eventos de Geofencing (*UserNearbyCanteensGeofencingBroadcastReceiver*).

Adicionalmente às três packages principais previamente descritas, foi também concebido uma quarta package, de nome *util*, que tem como responsabilidade dar acesso a um conjunto de funções e classes que auxiliam as restantes unidades de implementação. Nesta package pode ser encontra classes como:

- *Client*, que permite realizar um pedido HTTP, retornando a resposta deste como um objeto *Request*, que expõem o *status-code* e *payload* da resposta do pedido. Estas classes foram concebidas de raiz, pois na opinião dos desenvolvedores, a utilização de uma biblioteca externa que permite o mesmo só é vantajosa se for necessária a utilização dos padrões *interceptor* e *middleware.*
- *CommunicationMediator*, que atua como um intermediário no acesso do estado de vários serviços de comunicação (e.g. verificar se existe conexão à internet)
- *Provider*, que fornece pontos de entrada únicos a diversos objetos cujo estado pode vária com o decorrer da aplicação (e.g. RepositoryFactory), sendo este a entidade que tem a responsabilidade gerir o estado destes mesmos.
- *Settings*, que unifica todas as definições existentes na aplicação (e.g. saber se a aplicação está em modo offline, a utilizar o modo escuro). Esta entidade utiliza as *SharedPreferences* [15] do dispositivo como mecanismo de persistência das definições, e coloca em memória o valor destas (mecanismo caching), de modo a reduzir os acessos à memoria interna do dispositivo.

5.2 Princípios de Software Seguidos

Tendo por base a aplicação dos padrões de design e software definidos, foi também planeado o seguimento de diversos princípios de software de modo a permitirem um melhor fluxo de desenvolvimento da aplicação e reduzir possíveis custos de *refactoring* futuros. Os dois maiores princípios que a aplicação segue são:

- *Single Responsibility Principle* (SRP) – De modo a reforçar a separação de responsabilidades sugerida pelo padrão MVP, todas as unidades de implementação (métodos, classes, packages) foram minimizadas ao máximo de uma responsabilidade.
- *Intention Revealing Interface (IRI)* – A identificação das unidades de implementação com identificadores que expõem o valor produzido, pertencentes à linguagem ubíqua do domínio correspondente, permitem a redução de comentários que explicam o que estas fazem, e dão logo a conhecer ao desenvolvedor a/s sua/s responsabilidade/s. O seguimento deste princípio complementa o SRP.

5.3 Técnicas de Controlo do Processamento e Memória Utilizada

De modo a reduzir o processamento necessário na *main/ui thread* da aplicação, todas as operações que pudessem ser demoradas ou pesadas (e.g. comunicação à Internet, leitura e escrita na memória interna do dispositivo), foram executadas numa thread separada. Para tal, foram utilizadas as *AsyncTask* do Android, que providenciam um conjunto de *callbacks* de modo a controlar o estado da tarefa. Sempre que necessário a utilização das AsyncTask, estas foram declaradas como uma classe privada à classe em que estão a ser utilizadas, de modo a expor a sua responsabilidade e intenção (SRP + IRI), como também para aumentar a legibilidade e compreensão do código. As únicas exceções onde não foram criadas AsyncTask para operações potencialmente pesadas, residem na comunicação com as

SharedPreferences do dispositivo, devido a API que fornece a comunicação com estas já criar internamente threads que gerem o acesso à memoria interna do dispositivo.

Também na gestão da memória utilizada e de forma a prever potenciais *memory leaks*, todos os objetos que estabelecem uma stream de comunicação (e.g. *LocationManager* para obtenção dos dados da localização em tempo real), foram *destruídos* no fim do ciclo da atividade em que reside (método onDestroy).

5.4 Estruturação da User-Interface

Visto que a propósito da aplicação consiste em permitir ao utilizador a consulta de dados relativos às ementas das cantinas das escolas do IPP, maioritariamente é necessário apresentar ao utilizador os dados de forma enumerada. Para tal recorreu-se à utilização de listas e ícones que identificam as respetivas propriedades da informação, guiando o utilizador pela aplicação e fornecendo uma melhor experiência de utilização. Um exemplo notório da utilização destes elementos gráficos reside na consulta dos pratos disponíveis numa determinada ementa, onde estes são enumerados numa lista, e identificados tanto visualmente, com a utilização de ícones que representam o seu tipo, como também textualmente através do reforço do tipo com a fonte *bold*. Ainda também com a utilização de ícones, o utilizador consegue verificar quais os pratos que selecionou como favorito, através da representação destes com uma estrela cheia (favorito) e não cheia (não marcado como favorito).

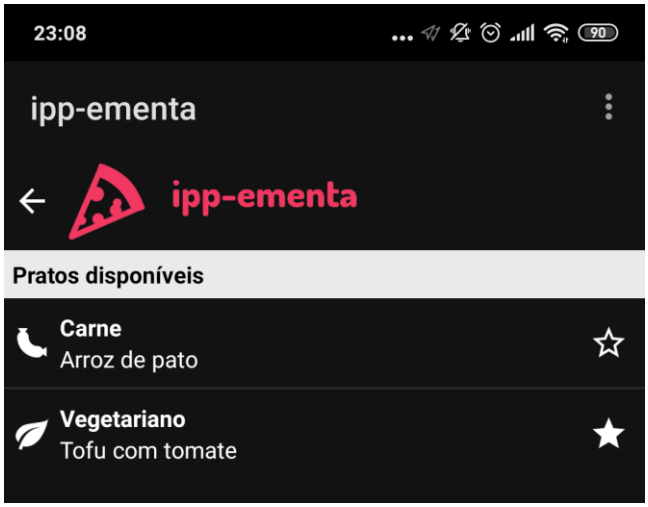

Figura 2 - Consulta dos pratos de uma ementa

Relativamente à estilização da user-interface, foi seguido os *design guidelines* do Material Design [16]. Para tal foi necessário a declaração da dependência dos componentes do Material Design, criar e declarar temas específicos no ficheiro *styles*.xml que extendem os do material design e por fim aplicar estes ao layout da atividade através do método *getTheme*.

6. REFERÊNCIAS

[1] F. & N. President's Council on Sports, «Importance of Good Nutrition», *HHS.gov*, 20-Jul-2012. [Em linha]. Disponível em: https://www.hhs.gov/fitness/eat-healthy/importance-of-good-nutrition/index.html. [Acedido: 29-Nov-2019].

[2] «The risks of poor nutrition :: SA Health». [Em linha]. Disponível em: https://www.sahealth.sa.gov.au/wps/wcm/connect/public+content/sa+health+internet/healthy+living/is+your+health+at+risk/the+risks+of+poor+nutrition. [Acedido: 29-Nov-2019].

[3] «Consumos e Estilos de Vida no Ensino Superior». .

[4] P. Emanuel e J. Freitas, «Aplicações Móveis no Mundo Universitário», p. 4.

[5] «euIPP – Aplicações no Google Play». [Em linha]. Disponível em: https://play.google.com/web/store/apps/details?id=com.moofwd.ipporto&hl=pt_PT. [Acedido: 29-Nov-2019].

[6] «Distribution dashboard», *Android Developers*. [Em linha]. Disponível em: https://developer.android.com/about/dashboards. [Acedido: 30-Nov-2019].

[7] mapsforge, *mapsforge/mapsforge*. 2019.

[8] «OpenStreetMap», *OpenStreetMap*. [Em linha]. Disponível em: https://www.openstreetmap.org/about. [Acedido: 30-Nov-2019].

[9] «Firebase Cloud Messaging | Firebase». [Em linha]. Disponível em: https://firebase.google.com/docs/cloud-messaging. [Acedido: 30-Nov-2019].

[10] «Firebase Pricing | Firebase». [Em linha]. Disponível em: https://firebase.google.com/pricing. [Acedido: 30-Nov-2019].

[11] «SQLite Home Page». [Em linha]. Disponível em: https://www.sqlite.org/index.html. [Acedido: 30-Nov-2019].

[12] «Localize your app», *Android Developers*. [Em linha]. Disponível em: https://developer.android.com/guide/topics/resources/localization. [Acedido: 30-Nov-2019].

[13] «Room Persistence Library», *Android Developers*. [Em linha]. Disponível em: https://developer.android.com/topic/libraries/architecture/room. [Acedido: 30-Nov-2019].

[14] «Save data in a local database using Room», *Android Developers*. [Em linha]. Disponível em: https://developer.android.com/training/data-storage/room. [Acedido: 30-Nov-2019].

[15] «SharedPreferences | Desenvolvedores Android», *Android Developers*. [Em linha]. Disponível em: https://developer.android.com/reference/android/content/SharedPreferences?hl=pt-br. [Acedido: 05-Jan-2020].

[16] «Design», *Material Design*. [Em linha]. Disponível em: https://material.io/design/. [Acedido: 05-Jan-2020].