P24–28-Visualização da Ocupação de Salas em Dispositivos Móveis através da Plataforma de Gestão de Recursos da UTAD

Renato Manuel Costa Moreira Maia Mendes (65384), João Paulo Moura (UTAD/ECT)

***Abstract*—The present article presents the development of a prototype system to visualize the classroom occupancy in the University of Trás-os-Montes and Alto Douro (UTAD). The aim of the system is to optimize the usage of the physical spaces in the Institution, improving the display of the classroom's schedule and subsequent scheduling. The prototype was developed with modern and robust technologies, such as Android, Kotlin, Jetpack Compose, Room, Retrofit, Hilt and RESTfulAPIs, a multi-module architecture and possess a intuitive graphic interface to make it accessible for the academic community to use. The design and implementation of the system were planned thoroughly by a requirement specification with resource of diagrams UML, in order to obtain accurate results. The end result show us the system's potential as well has the successfully integration with the UTAD's resource management platform, ensuring reliability and scalability in the proposed solution, which is geared towards the digital transition.**

***Index Terms*—Digital, Room Management, Multi-module, System, Android.**

I. Introdução

A gestão eficiente dos espaços físicos é crucial para o bom funcionamento de qualquer organização e a gestão dos espaços académicos em instituições de ensino tem-se mostrado frequentemente ineficaz perante as necessidades da comunidade educacional. O modelo tradicional, maioritariamente baseado em métodos manuais e desatualizados, enfrenta diversos desafios e encontra-se desfasado da realidade, tornando-se rudimentar e insustentável.

Salas de aula, laboratórios, auditórios e outros espaços académicos representam recursos valiosos para as organizações de ensino, que devem ser utilizados de forma competente para atender à procura da comunidade académica em constante expansão. Impraticabilidade e burocracias nos processos de reserva de salas e subutilização das mesmas são alguns dos problemas que persistem no sistema tradicional e que o tornam ineficiente.

Estando na era da informação, a transição digital assume-se como ferramenta crucial estratégica tanto no desenvolvimento de uma organização, como no desenvolvimento de um país. Em Portugal, adotou-se, a 21 de Abril de 2020, o Plano de Ação para a Transição Digital (PATD), com vista numa sociedade portuguesa digital[1-2]. Este plano foi desenvolvido em articulação com os incentivos da União Europeia (UE) e com o Plano de Ação para a Educação Digital (2021-2027) que a Comissão Europeia viria a lançar a 30 de Setembro de 2020, com ambição numa educação europeia moderna e digital de excelência[3].

A Universidade de Trás-os-Montes e Alto Douro (UTAD), em Portugal, é uma das instituições de ensino que atravessa um processo de transformação digital e que visa, efetivamente, tornar-se mais digital, através de investimentos estratégicos e da adoção de tecnologias inovadoras que possam aprimorar metodologias internas, melhorar a experiência académica e fortalecer a sua posição como referência no panorama educacional português.

Não é novidade que a UTAD já há muitos anos que provém de ferramentas informáticas, como o Sistema de Informação de Apoio ao Ensino (SIDE), para elaboração de horários e conciliação de recursos[4]. Contudo, face às novas exigências educativas e planos formativos gradativamente elaborados por módulos e dependências a recursos específicos, tornou-se imperativo encontrar uma alternativa especializada na gestão de horários e recursos.

A Bullet Solutions é uma empresa portuguesa com provas dadas a nível internacional, que oferece uma plataforma robusta e flexível de gestão e aproveitamento eficiente de recursos, adaptável às necessidades específicas das organizações de ensino[5].

Nesse contexto e tentando potenciar o aproveitamento de recursos e automatizar os processos da instituição, a UTAD adquiriu, recentemente, os serviços desta empresa[6].

Em articulação com esta integração, propõe-se a implementação de um sistema protótipo de software dinâmico que facilite e potencialize o acesso às funcionalidades fornecidas pela plataforma da Bullet, entre elas: apresentação da ocupação das salas atualizadas e reserva de slots de salas disponíveis, através de dispositivos móveis no local, que atendem as necessidades da comunidade académica da UTAD.

Este sistema visa agilizar o processo de visualização de ocupação eficiente de salas e o seu agendamento, em contexto universitário, através de uma aplicação interativa para tablets, posteriormente instalados nas portas das salas. A interface gráfica do sistema propõe ser simples e intuitiva e a sincronização com a plataforma da Bullet Solutions garantirá a validação de disponibilidade e permissões de reservas.

João Paulo Moura pertence ao Departamento de Engenharias da Escola de Ciências e Tecnologia da Universidade de Trás-os-Montes e Alto Douro e é Investigador Sénior do INESC-TEC (ver perfil ORCID em https://orcid.org/0000-0002-4543-0237.

Artigo submetido em 26 de Julho de 2024.

II. Enquadramento Tecnológico

O protótipo proposto assenta em tecnologias modernas e robustas. Um sistema Android projetado para tablets que tem como finalidade a visualização do cronograma das salas e a possibilidade de ocupação das mesmas nos períodos em que estão disponíveis. Desta forma, os utilizadores poderão fazer as suas próprias reservas, o que simplifica os processos de acesso à plataforma de gestão de recursos da Universidade e promove uma experiência agradável, interativa e responsiva no local.

Tendo em consideração a aplicabilidade, escalabilidade, performance e as melhores práticas de desenvolvimento de software definiu-se a seguinte abordagem:

A. Sistema Operativo

Com vista numa aplicação móvel com alta compatibilidade com tablets, optou-se pelo **Android** como sistema operativo[7-8].

Desenvolvido e atualizado regularmente pela Google, o Android fornece um conjunto robusto de ferramentas de desenvolvimento, como o Android Studio, e de materiais de suporte com uma vasta comunidade de *developers*, que facilita o processo de criação e desenvolvimento de aplicações[9]. Além disso, a Google Play Store, uma loja global de aplicações com milhões de utilizadores, oferece a plataforma ideal para distribuição e acesso à aplicação que se pretende implementar.

B. Linguagem de Programação

De acordo com o sistema operativo, decidiu-se que a aplicação seria desenvolvida em **Kotlin**, a linguagem de programação oficial do Android[10].

Linguagem relativamente moderna, com uma sintaxe expressiva, concisa e segura, recomendada pela Google para o desenvolvimento de aplicações móveis. Oferece recursos avançados como programação orientada a objetos com segurança contra null e funções de extensão, interoperabilidade com Java, e corrotinas, para um código assíncrono e seguro contra thread, ideal para operações de rede e de base de dados. Recentemente, proporciona também o desenvolvimento multiplataforma, não estando limitada unicamente ao sistema operativo Android[11].

C. Framework UI

Relativamente à interface e ao design da aplicação e em articulação com o sistema operativo Android e a linguagem Kotlin, escolheu-se o **Jetpack Compose**.

Toolkit moderno e declarativo para o desenvolvimento nativo de interfaces gráficas Android[12]. Baseado em Kotlin e desenvolvido pela Google, oferece simplicidade, performance e flexibilidade de código adaptável a diferentes dispositivos e tamanhos de tela, permite desenvolver interfaces fluidas, com alto desempenho e com menos código, proporcionando uma ótima experiência tanto para os programadores como para os utilizadores.

D. Serviços Web

De forma a mediar e facilitar o acesso e o aproveitamento dos recursos e das funcionalidades da plataforma da Bullet Solutions, integrou-se uma **RESTfulAPI**[13], também fornecida pela empresa, designada BulletAPI[14]. Através do gateway da API é possível beneficiar dos microserviços que a empresa fornece.

E. Bibliotecas/Ferramentas de Desenvolvimento

Em resposta a certas particularidades no desenvolvimento de aplicações como comunicação com APIs, persistência de dados, injeção de dependências, entre outras, recorreu-se a algumas bibliotecas com funcionalidades específicas de desenvolvimento projetadas para a integração com o Android:

*1) **Retrofit**:* Simplifica a comunicação com APIs RESTful, permitindo realizar requisições HTTP de forma eficiente[15]. Essencial na mediação com a API da Bullet Solutions, mostrada na figura 1.

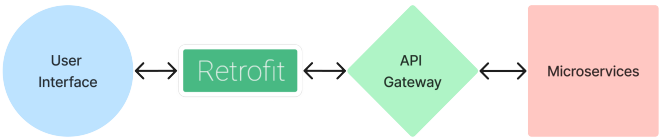

Fig. 1: Fluxograma de utilização da biblioteca Retrofit.

*2) **Room**:* Facilita a persistência e a abstração de dados provenientes da API numa base de dados SQLite[16]. Ideal para aplicações Android que priorizam o modo offline[17]. Na figura 2 observa-se o seu fluxo.

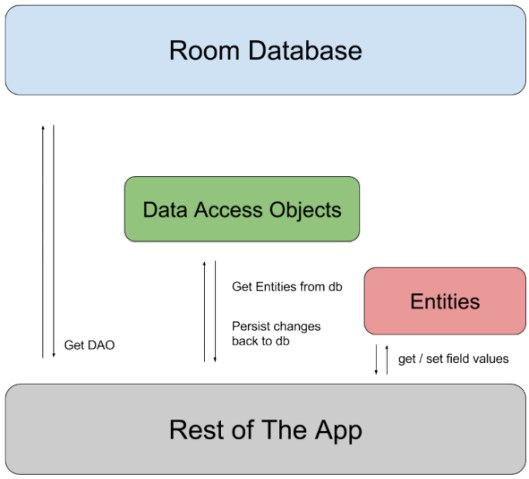

Fig. 2: Diagrama da arquitetura da biblioteca Room.

*3) **Hilt**:* Gere as dependências da aplicação de forma eficaz e escalável, reduz o código repetitivo e facilita a modularização[18].

F. Arquitetura

Na perspetiva de desenvolvimento de uma aplicação robusta, testável e de fácil manutenção, decidiu-se dividir o sistema em módulos autônomos e reutilizáveis e adotar uma **arquitetura**

modular e por camadas[19-20]. Na figura seguinte destaca-se a arquitetura típica recomendada pelo Android com princípios como separação de conceitos, fonte de verdade única, fluxo de dados unidirecional e UI baseada em modelos de dados[21].

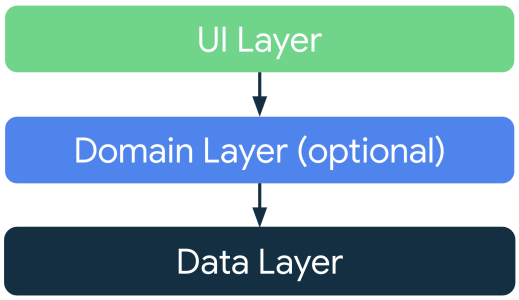

Fig. 3: Arquitetura típica de uma aplicação Android.

III. Conceção

Através de um processo de conceção focado nas funcionalidades requeridas e nas necessidades da comunidade académica, recorreu-se à plataforma da Bullet Solutions e realizou-se o levantamento de requisitos para uma modelagem clara, abrangente e precisa do sistema.

Destaca-se a articulação de dois tipos diagramas UML[22] para cada requisito:

1) **Diagramas de Caso de Uso:** Definem as funcionalidades e os atores do sistema, as suas interações e estabelecem limites no domínio das aplicabilidades do sistema e das responsabilidades dos atores.
2) **Fluxogramas de Atividades:** Descrevem o fluxo de atividades para cada processo ou caso de uso, identificando os fluxos de trabalho e as dependências entre atividades, facilitando a compreensão do funcionamento interno do sistema e as ações necessárias para alcançar um objetivo, por ordem que são executadas.

De modo a representar os utilizadores ou entidades externas que interagem com o sistema, estipulou-se quatro tipos de atores:

- **Administrador (A)** - Utilizador com privilégios administrativos que configura o sistema.
- **Utilizador Convidado (GU)** - Utilizador que pode visualizar o horário da sala sem necessidade de autenticação.
- **Utilizador Autenticado (AU)** - Utilizador que pode realizar reservas, editar e cancelar.
- **API** - Interface de programação da Bullet Solutions que permite a interoperabilidade com a plataforma de gestão de recursos da universidade.

Definiram-se, cinco requisitos de alto nível para o sistema:

A. Configuração de um dispositivo móvel para apresentação da ocupação de uma sala

Requisito de atribuição de dispositivos a salas, ativas, na plataforma de gestão de recursos da UTAD, repensentado pelas figuras 4 e 5.

Concede ao Administrador a responsabilidade de associar e configurar determinado dispositivo móvel a uma sala específica, garantindo a apresentação do cronograma da sala atualizado e o acesso a recursos móveis de reserva.

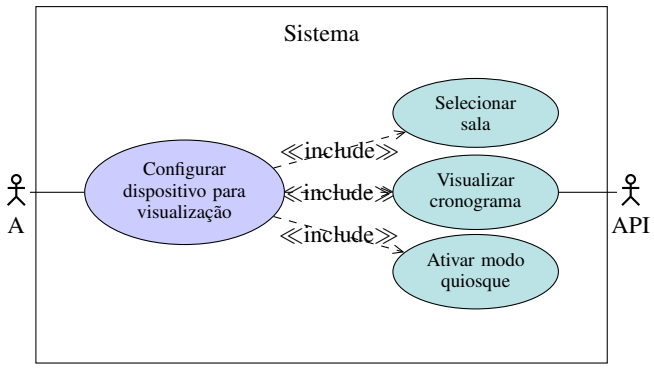

Fig. 4: Diagrama de caso de uso referente ao 1º requisito.

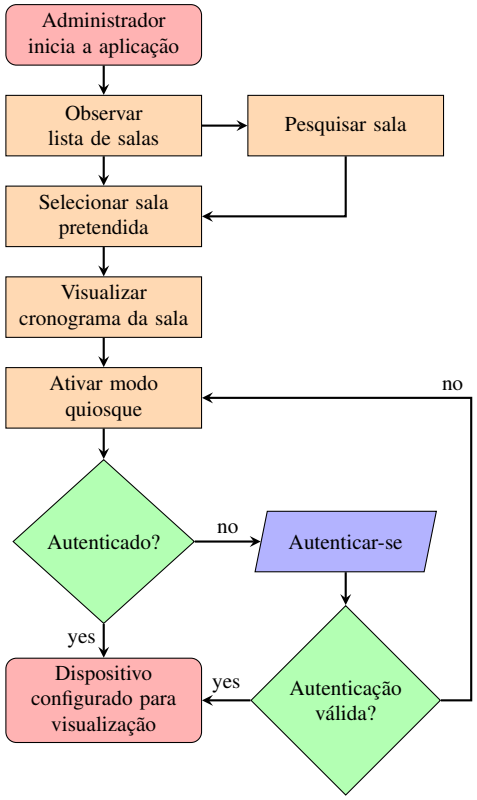

Fig. 5: Fluxograma de atividades referente ao 1º requisito.

B. Visualização da ocupação de uma sala

Requisito de exibição do horário de uma sala, retratado nas figuras 6 e 7. Abrange dois tipos de utilizadores, convidado e autenticado, e permite a alteração do tipo de vista do cronograma entre diário/semanal e a alteração da data.

Uma vez que a ocupação de salas está dependente da plataforma de gestão de recursos académicos da universidade, o cronograma está suscetível a alterações/adições de eventos e, portanto, é necessário que haja uma atualização automática

dos mesmos através da API. Deste modo, os utilizadores podem visualizar a versão mais recente dos horários das salas, mostrando se a sala está ocupada ou disponível de forma precisa.

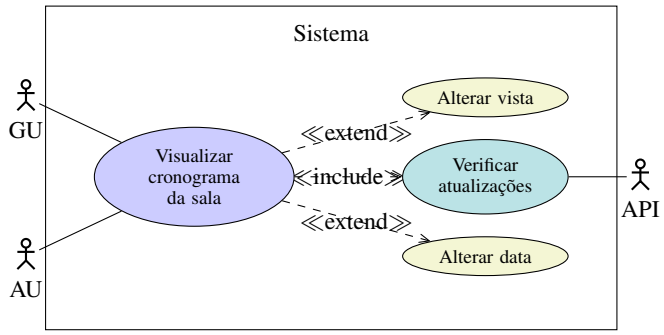

Fig. 6: Diagrama de caso de uso referente ao 2º requisito.

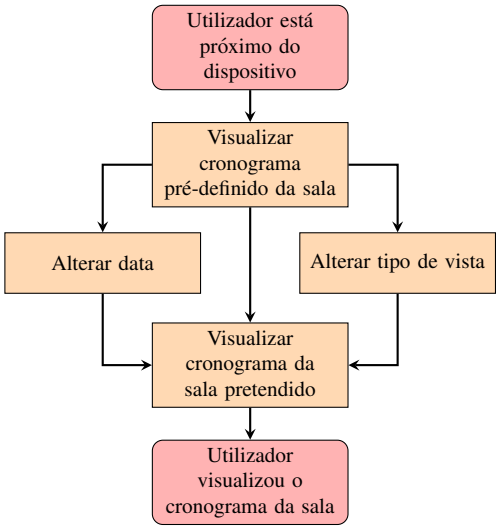

Fig. 7: Fluxograma de atividades referente ao 2º requisito.

C. Reserva de um período disponível de uma sala

Definiu-se como terceiro requisito do sistema o processo de reserva de uma sala, caraterizado pelas figuras 8 e 9.

Concede aos utilizadores autenticados a vantagem de poderem reservar uma sala disponível para o período de tempo que pretendem.

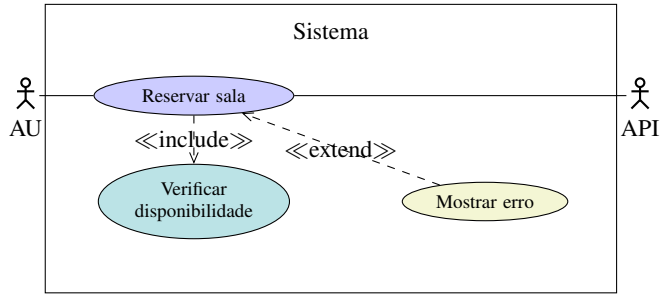

Fig. 8: Diagrama de caso de uso referente ao 3º requisito.

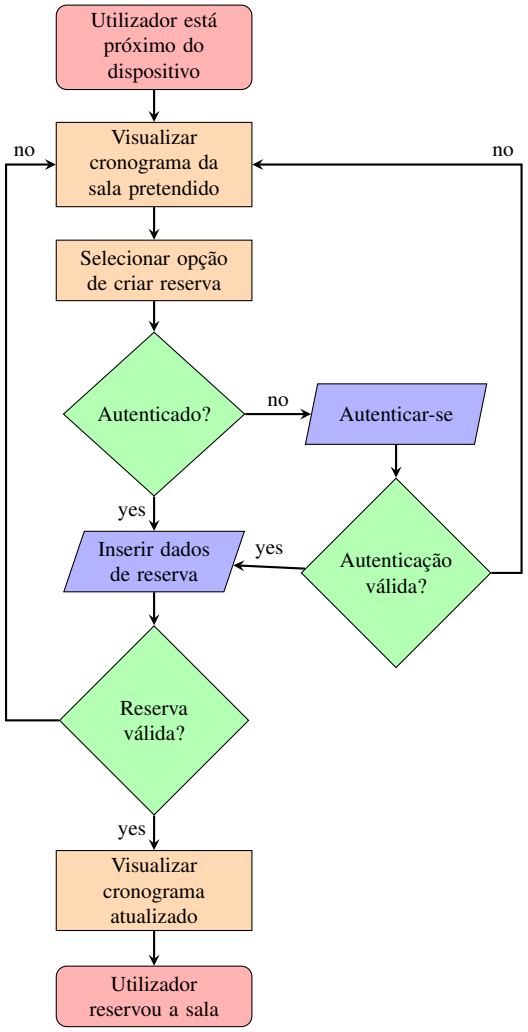

Fig. 9: Fluxograma de atividades referente ao 3º requisito.

D. Cancelar uma reserva de uma sala

Com o aumento da necessidade de adaptabilidade e agilidade em ambientes dinâmicos e colaborativos, o cancelamento de reservas de salas torna-se um requisito essencial para o sistema.

Caracterizado nas figuras 10 e 11, fornece-se aos utilizadores autenticados a possibilidade de cancelar uma reserva previamente feita, através de uma interação rápida com o sistema, de modo a evitar indisponibilidades desnecessárias e maximizando o aproveitamento do espaço.

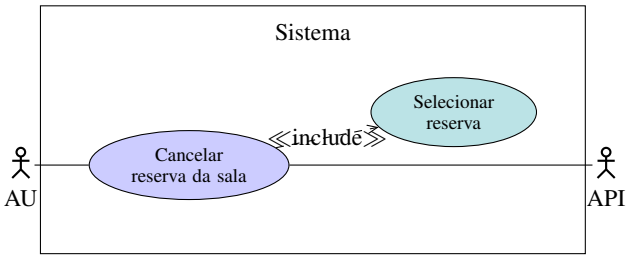

Fig. 10: Diagrama de caso de uso referente ao 4º requisito.

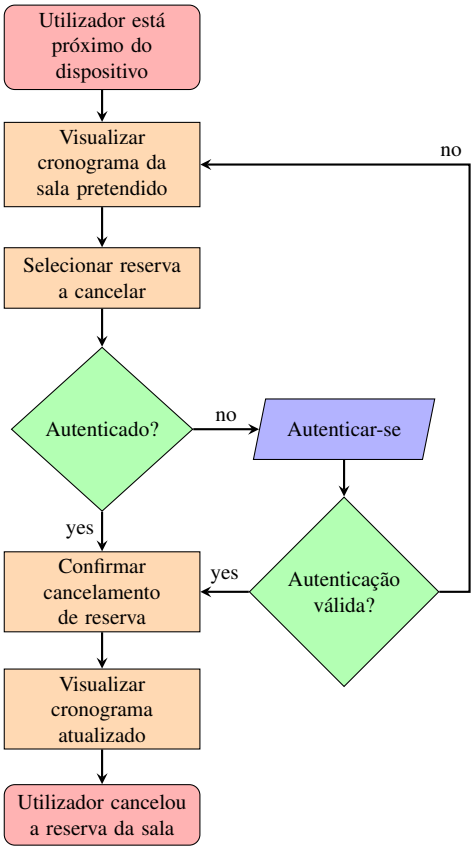

Fig. 11: Fluxograma de atividades referente ao 4º requisito.

E. Edição de uma reserva para um período disponível de uma sala

De forma a conceder flexibilidade aos utilizadores de acordo com as suas necessidades, incluiu-se no sistema o requisito de edição de reservas, representado nas figuras 12 e 13.

Concede aos utilizadores autenticados, a vantagem de modificar uma reserva previamente feita. Desse modo, o utilizador pode alterar os detalhes da reserva (data, hora, nome do evento, etc.) e submeter as alterações da reserva.

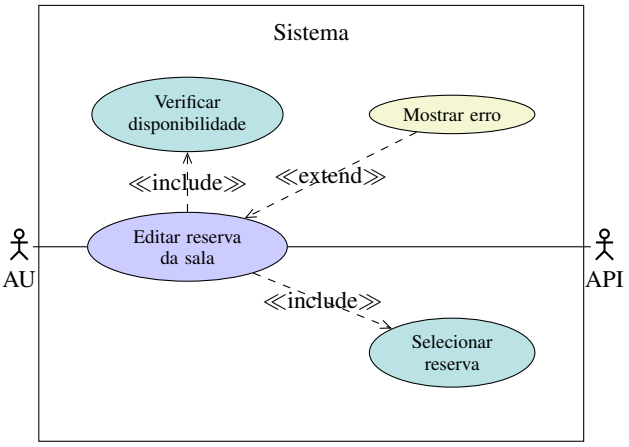

Fig. 12: Diagrama de caso de uso referente ao 5º requisito.

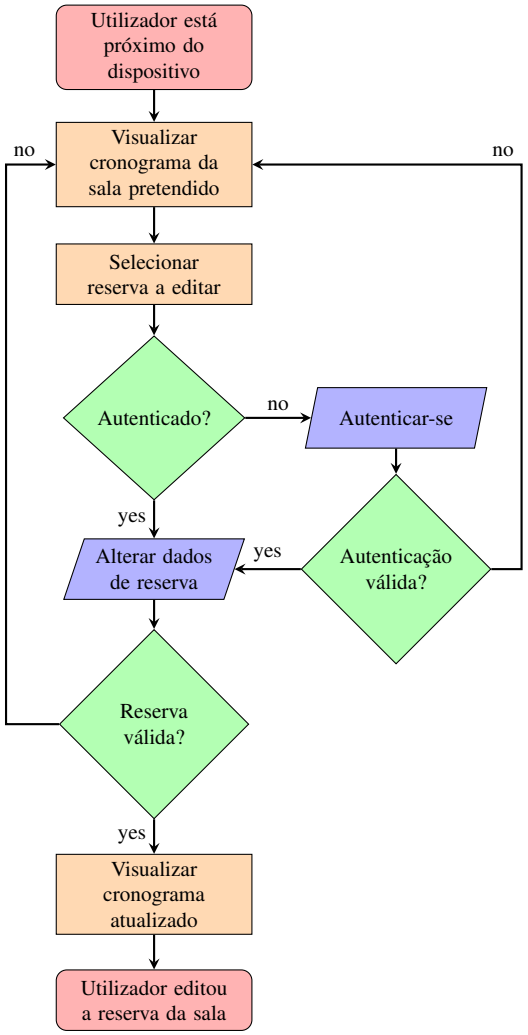

Fig. 13: Fluxograma de atividades referente ao 5º requisito.

IV. Implementação

Com base nas tecnologias estipuladas e na conceção do sistema prosseguiu-se com a sua implementação.

Como já referido, projetou-se, para o sistema, uma arquitetura modular. De forma a identificar, corretamente, os principais componentes do sistema, as suas interações e definir os módulos, em concordância com os requisitos pré-estabelecidos, decidiu-se, primeiramente, avaliar e testar a plataforma de gestão de recursos da UTAD e a respetiva API, fornecidas pela Bullet Solutions.

A. Dados

Uma vez que se pretende visualizar a ocupação das salas da universidade e posteriormente fornecer funcionalidades de reserva, definiu-se como recursos essenciais a representar, no sistema, as salas ativas e os respectivos eventos, mostrado na figuras 14.

Para efeitos de complementaridade e modelagem correta dos recursos, recorreu-se também a alguns modelos de suporte da API.

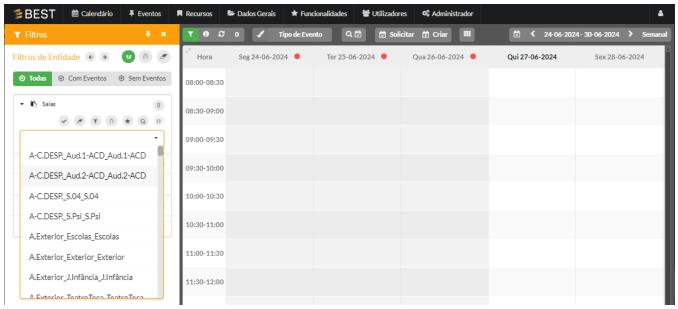

Fig. 14: Plataforma da gestão de recursos da UTAD.

B. Funcionalidades

De forma a identificar corretamente as funcionalidades da plataforma de gestão de recursos da universidade, estipulou-se, como sala de teste, a sala F1.01 do polo 1 da Escola de Ciências e Tecnologias da UTAD. Assim, verificou-se a concordância com os requisitos pré-estabelecidos e definiu-se as seguintes funcionalidades:

- Visualizar o cronograma da sala;

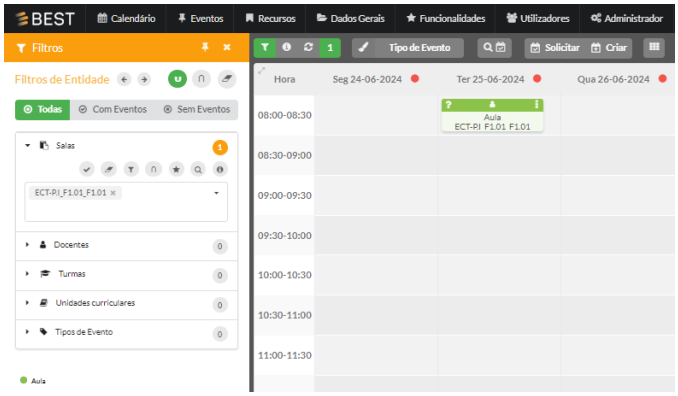

Fig. 15: Visualização da ocupação de uma sala da UTAD na plataforma da Bullet.

- Reservar a sala

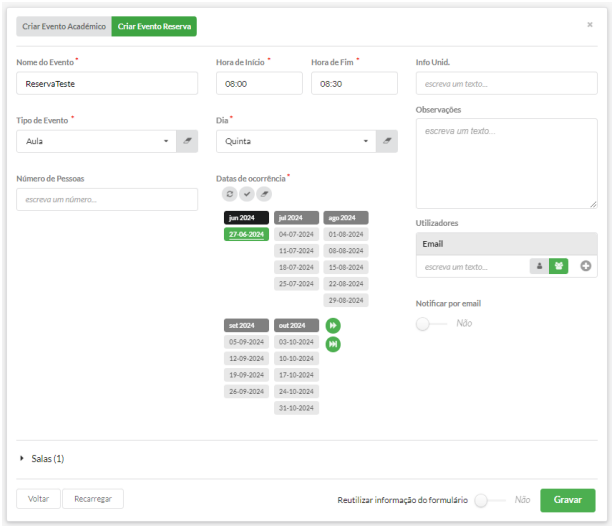

Fig. 16: Criação de uma reserva na plataforma da Bullet.

- Editar a reserva da sala

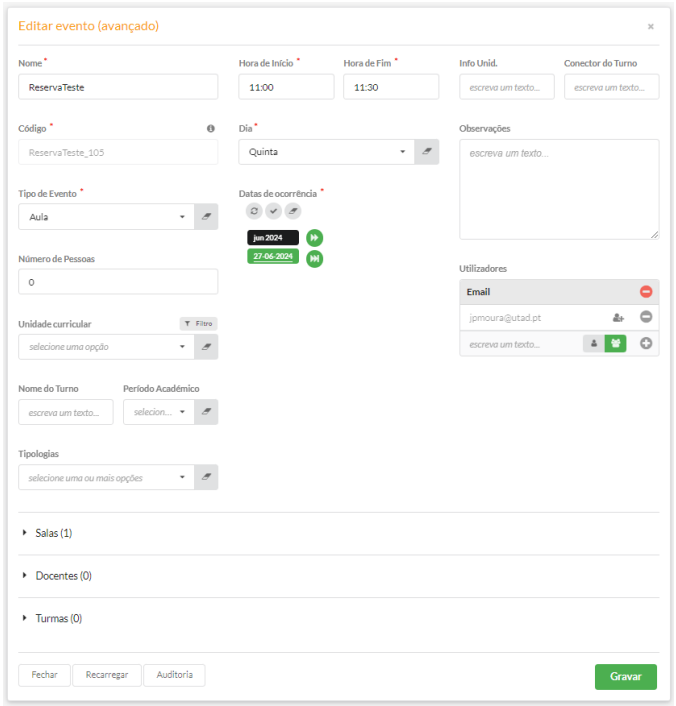

Fig. 17: Edição de uma reserva na plataforma da Bullet.

- Cancelar a reserva da sala

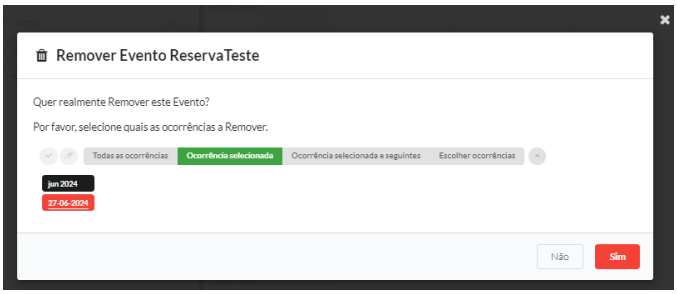

Fig. 18: Cancelamento de uma reserva na plataforma da Bullet.

Estabelecidos os dados e as funcionalidades requeridas, identificaram-se os endpoints necessários da API, presentes na seguinte figura.

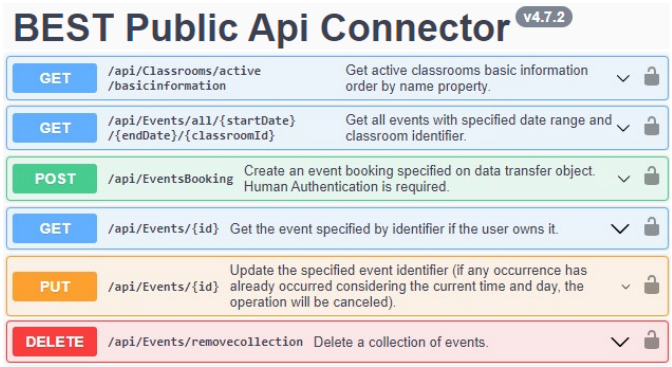

Fig. 19: Endpoints identificados na API da Bullet.

C. Módulos

De seguida, estruturou-se a arquitetura do sistema da seguinte forma:

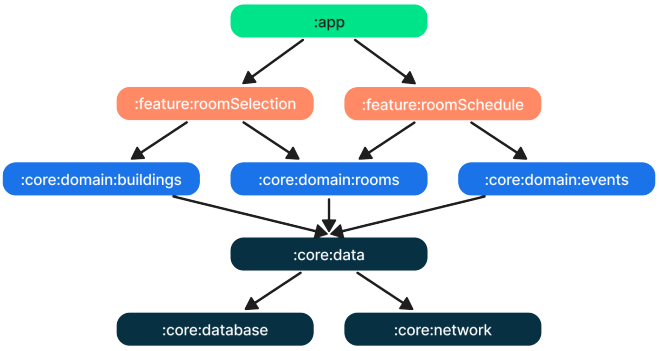

Fig. 20: Estrutura da arquitetura modular definida.

Procedendo-se à sua implementação, surgiu a necessidade de se incluir alguns módulos de suporte, que podem ser observados na figura 21.

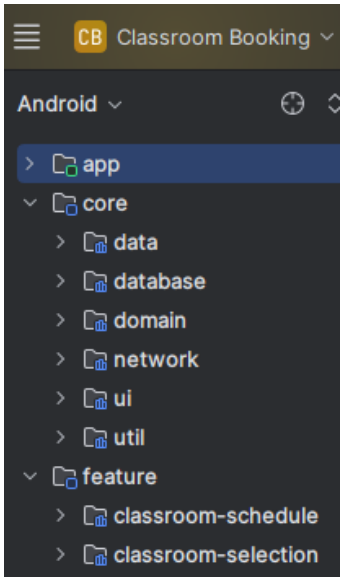

Fig. 21: Estrutura do projeto multi-módulo implementada.

A ordem de implentação foi a seguinte:

1) Módulo de rede: Integra os endpoints e os modelos de dados remotos da Bullet Api, através da biblioteca Retrofit, demonstrado no anexo A.

Por questões de segurança, as aplicações, por defeito, não possuem conectividade com a Internet. De forma a permitir que o sistema consiga comunicar com a Api, surgiu a necessidade de declarar, explicitamente, que a aplicação necessita de acesso à Internet, como se pode verificar na seguinte listagem.

Listagem 1: Declaração da permissão de acesso à Internet.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

2) Módulo de base de dados: Persiste os modelos de dados remotos provenientes da Api em entidades, de modo a que o sistema possa priorizar o modo offline.

Na figura 22 verifica-se a relação estabelecida entre as entidades:

- Um edifício possui múltiplas salas (1:N) e uma sala pode possuir múltiplos eventos (N:M).

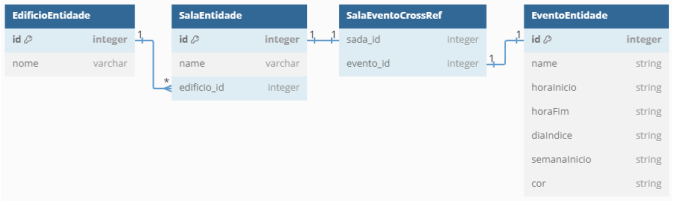

Fig. 22: Relação de entidades estipulada.

3) Módulo de domínio de dados: Representa os modelos de dados que vão interagir com a camada de apresentação do sistema, demonstrado na listagem seguinte.

Listagem 2: Modelos de dados requeridos para apresentação.

```
data class Building(
    val id: Int,
    val name: String
)
data class Classroom(
    val id: Int,
    val name: String,
    val buildingId: Int
)
data class Event(
    val id: Int,
    val name: String,
    val startDateTime: LocalDateTime,
    val endDateTime: LocalDateTime,
    val color: Long,
)
```

4) Módulo de abstração dados: Trata do mapeamento dos modelos de dados remotos e locais em modelos de domínio e da intermediação entre os dados e as funcionalidades, demonstrado no anexo B.

5) Módulos de funcionalidades: Com propósito de simplificar e agregar as funcionalidades do sistema definiram-se dois módulos de apresentação:

i. **Módulo de Seleção de Sala** - Módulo referente ao ecrã da configuração/atribuição da sala ao dispositivo, representado no anexo C.
ii. **Módulo de Apresentação da Ocupação da Sala** - Módulo referente ao ecrã da visualização do cronograma da sala e das funcionalidades de reserva.

6) Módulo da aplicação: O módulo da aplicação está encarregue da navegação entre as funcionalidades do sistema, caracterizado no anexo D. As dependências inter módulos foram mediadas pela biblioteca Hilt.

V. Resultados e Discussão

Durante a implementação do sistema, surgiram dois tipos de resultados, os resultados pré e pós implementação dos *feature modules*. De notar que, até à data, apenas as funcionalidades de seleção de sala e de apresentação de ocupação foram realizadas.

Os resultados intermédios estão presentes na seguinte figura.

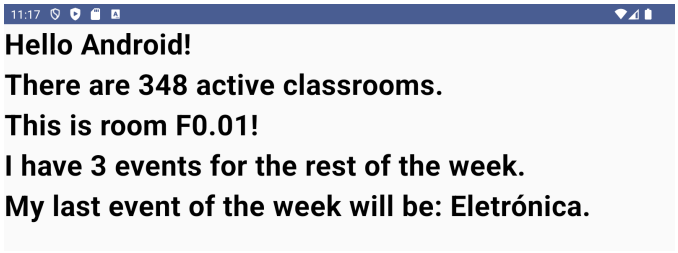

Fig. 23: Primeiro resultado obtido.

Estes resultados foram obtidos através da interação dos módulos de aplicação e de abstração de dados.

Com a implementação dos módulos de funcionalidades do sistema, e os respetivos ecrãs, obteve-se os seguintes resultados finais:

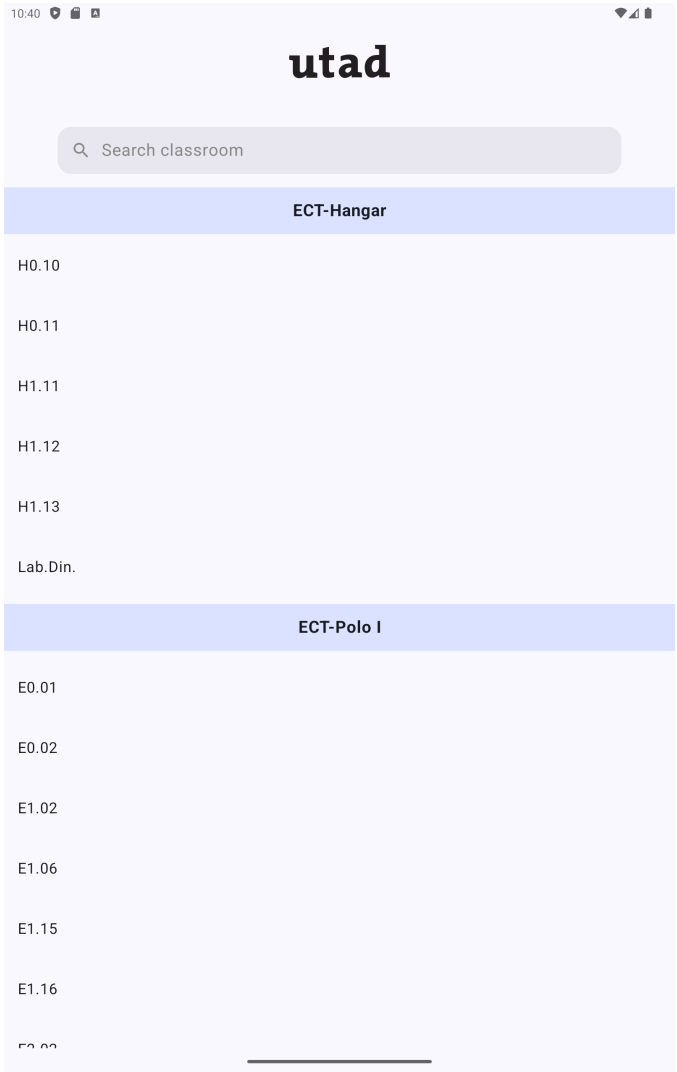

Fig. 24: Segundo resultado obtido.

A figura 24 retrata os resultados obtidos da comunicação entre o módulo de aplicação e o módulo de seleção de sala. Representa, somente, a funcionalidade de atribuição de sala, uma vez que, a funcionalidade de configuração do dispositivo, até ao momento, não foi implementada. A ferramenta de procura de sala, está integrada, mas ainda não se encontra em funcionamento.

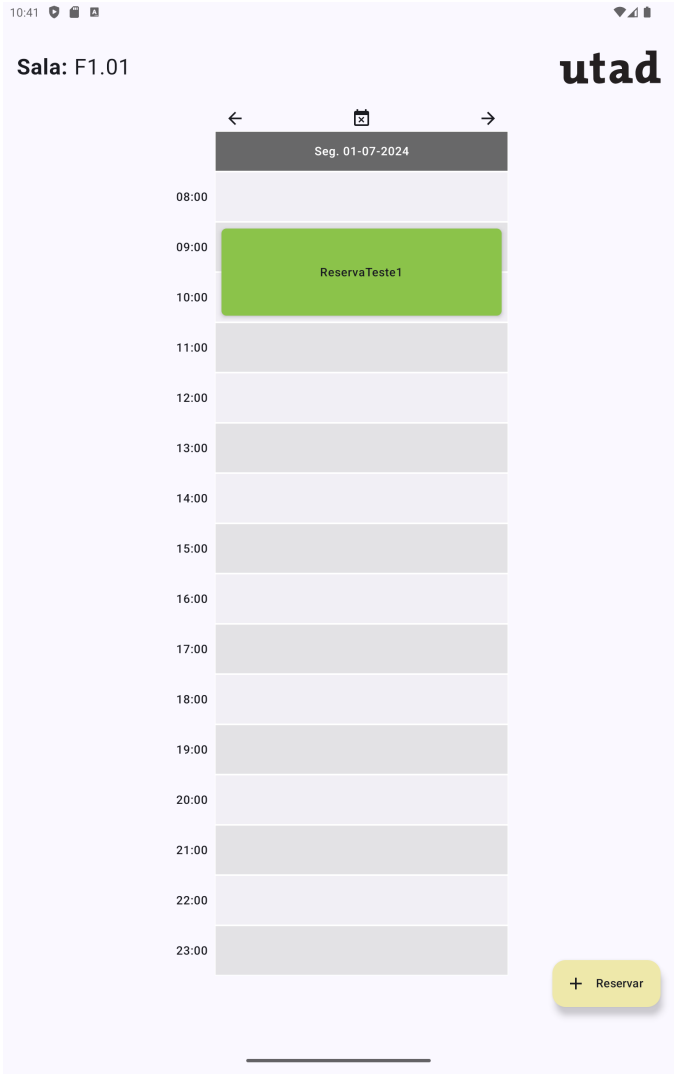

Fig. 25: Terceiro resultado obtido.

A figura 25 ilustra o primeiro resultado obtido da comunicação entre o módulo de aplicação e o módulo de apresentção de ocupação de sala. Representa, somente, a funcionalidade de visualização de cronograma semanal da sala pré-selecionada, uma vez que, as funcionalidades de reserva, até ao momento, não foram implementadas. As ferramentas de criação de reserva e de alteração de data estão integradas, mas ainda não se encontram em funcionamento.

Como se pode verificar, o ecrã está em modo retrato e apresenta a disponibilidade da sala para apenas o primeiro dia da primeira semana de Julho de 2024. Por questões de apresentabilidade, as reservas mostradas foram introduzidas através da plataforma de gestão de recursos da UTAD, uma vez que, o ano letivo 23/24 está a terminar e já não há eventos.

A funcionalidade de alteração de modo de vista está integrada e a funcionar e é ativa após rotação do dispositivo como se pode verificar na figura 26. O ecrã está em modo paisagem e exibe a ocupação da sala ao longo da semana.

Fig. 26: Quarto resultado obtido.

VI. Conclusões

O desenvolvimento do sistema protótipo de apoio à gestão de ocupação de salas na Universidade de Trás-os-Montes e Alto Douro (UTAD), revelou-se um projeto desafiante e enriquecedor, com resultados promissores que demonstram a viabilidade e o potencial da solução para otimizar o aproveitamento dos espaços físicos da instituição e contribuir para uma experiência académica mais eficiente e agradável.

Para além dos benefícios mencionados anteriormente, também é necessário realçar a importância deste projeto na jornada de transição digital da instituição que não só se alinha com tendências globais como também se torna uma referência no uso de tecnologias digitais e de otimização de processo internos, servindo de exemplo para outras organizações de ensino em Portugal.

Os resultados obtidos até ao momento indicam que o sistema foi implementado com sucesso e que o objetivo de visualização de ocupação de salas em dispositivos móveis através da plataforma de gestão de recursos da UTAD, foi cumprido eficazmente. De notar que a comunicação eficiente entre o sistema protótipo e os recursos da infraestrutura de ensino, só foi possível graças à integração bem-sucedida da API da Bullet Solutions.

A implementação do protótipo baseou-se em tecnologias modernas e robustas, como Android, Kotlin, framework UI e bibliotecas de desenvolvimento de software reconhecidas, garantindo a escalabilidade, performance e confiabilidade da solução proposta. A interface gráfica projetou-se para ser simples e intuitiva, facilitando a utilização por parte da comunidade académica.

É importante destacar que o sistema ainda está em fase de desenvolvimento e que a implementação de funcionalidades de configuração de dispositivos e de reserva estão planeadas para um futuro próximo, de modo a oferecer uma experiência completa aos utilizadores. A realização de testes rigorosos e a posterior implementação completa do sistema, aliada ao seu acompanhamento/melhoramento contínuo, fazem parte dos planos a longo prazo.

Em suma, acredita-se que o projeto proposto tem o potencial de se tornar uma ferramenta valiosa e inovadora para a UTAD, que não só contribuirá na transformação digital, como também contribuirá significativamente para a melhoria da gestão de recursos e permitirá atender às necessidades específicas do corpo docente e discente da instituição.

Anexo A

Listagem 3: Integração dos endpoints no módulo de rede.

```
interface BulletApiServices {

    @GET("api/Classrooms/active/basicinformation")
    suspend fun getClassroomsActiveBasicInformation
    (): NetworkResponse<List<ClassroomDto>>

    @GET("/api/Events/all/{startDate}/{endDate}/{
    classroomId}")
    suspend fun getEventsByClassroomIdAndDateRange(
        @Path("startDate") startDate: String?,
        @Path("endDate") endDate: String?,
        @Path("classroomId") classroomId: Int?,
    ): NetworkResponse<List<EventDto>>

    @POST("/api/EventsBooking")
    suspend fun createEventBooking(@Body
    eventEditionUserAgreementsDTO:
    EventEditionUserAgreementsDTO?): NetworkResponse
    <EventEditionUserAgreementsDTO>

    @GET("/api/Events/{id}")
    suspend fun getEventById(
        @Path("id") id: Int?
    ): NetworkResponse<EventDto>

    @PUT("/api/Events/{id}")
    suspend fun updateEventById(
        @Path("id") id: Int?,
        @Body eventEditionUserAgreementsDTO:
    EventEditionUserAgreementsDTO?
    ): NetworkResponse<EventEditionUserAgreementsDTO
    >

    @DELETE("/api/Events/removecollection")
    suspend fun deleteEvent(
        @Body collectionIntDTO: CollectionIntDTO?
    ): NetworkResponse<ColletionIntDTO>
}
```

Anexo B

Listagem 4: Abstração e acesso aos dados.

```
class ClassroomRepositoryImpl @Inject constructor(
    private val classroomDAO: ClassroomDAO,
    private val buildingDAO: BuildingDAO,
    private val eventDAO: EventDAO,
    private val remoteDataSourceRepository:
    RemoteDataSourceRepository
): ClassroomRepository {
    override suspend fun upsertClassrooms() {
        // TODO() Synchronizer
        val networkClassrooms =
    remoteDataSourceRepository.getClassrooms()
        val networkBuildings = networkClassrooms.map
     { classroomDto ->
            classroomDto.building!!
        }.distinctBy { buildingDto ->
            buildingDto.id
        }
        upsertBuildings(networkBuildings)

        return classroomDAO.upsertClassrooms(
    networkClassrooms.map(ClassroomDto::
    asClassroomEntity))
    }
```

```
    override suspend fun upsertBuildings(buildings: List<BuildingDto>) {
        return buildingDAO.upsertBuildings(buildings.map(BuildingDto::asBuildingEntity))
    }

    override fun getBuildingsWithClassrooms(): Flow<List<BuildingWithClassrooms>> {
        return buildingDAO.getBuildingsWithRooms().map { it.map(BuildingWithClassroomsEntity::asBuildingWithClassrooms) }
    }
    //...
    override suspend fun upsertEvents(events: List<EventDto>) {
        return eventDAO.upsertEvents(events.map(EventDto::asEventEntity))
    }

    override suspend fun upsertEventCrossRef(classroomId: Int, startDate: String, endDate: String) {
        // TODO() Synchronizer
        val networkEvents = remoteDataSourceRepository
            .getEventsByClassroomIdAndDateRange(
                classroomId = classroomId,
                startDate = startDate,
                endDate = endDate,
            )
        upsertEvents(networkEvents)
        return classroomDAO.upsertEventCrossRefs(networkEvents.map(EventDto::toEventCrossRefs))
    }

    override fun getClassroomWithEvents(classroomId: Int): Flow<ClassroomWithEvents> {
        return classroomDAO.getClassroomWithEvents(classroomId).map { it.asClassroomWithEvents() }
    }
}
```

ANEXO C

Listagem 5: Implementação do ecrã referente ao módulo de seleção de sala.

```
@Composable
fun ClassroomSelectionScreen(
    state: ClassroomSelectionUiState,
    onClassroomClick: (classroom: Classroom) -> Unit,
){
    Column (
        modifier = Modifier.fillMaxSize()
    ) {
        Spacer(Modifier.height(16.dp))
        Image(
            imageVector = ImageVector.vectorResource(id = R.drawable.logo_utad_preto),
            contentDescription = "Logo",
            modifier = Modifier
                .width(120.dp)
                .aspectRatio(2f / 1f)
                .align(Alignment.CenterHorizontally),
        )
        Spacer(Modifier.height(16.dp))
        SearchBarUI(
            placeholder = "Search classroom",
        )
        Spacer(Modifier.height(16.dp))
        LazyColumnWithHeaders(
            list = state.buildingsWithClassrooms,
            onItemClick = {
                onClassroomClick(it)
            }
        )
    }
}
```

ANEXO D

Listagem 6: Integração da navegação entre os módulos de funcionalidades.

```
@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        //...
            SetupNavGraph(navController = navController, modifier = Modifier.padding(it))
    }
}

@Composable
fun SetupNavGraph(
    navController: NavHostController,
    modifier: Modifier = Modifier
){
    NavHost(
        navController = navController,
        startDestination = ClassroomSelection,
        modifier = modifier
    ) {
        classroomSelectionScreen(
            onClassroomClick = {
                navController.navigateToSchedule(it.id)
            }
        )
        classroomScheduleScreen()
    }
}
```

AGRADECIMENTOS

Os autores agradecem à empresa Bullet Solutions os meios colocados à disposição e que permitiram levar o trabalho apresentado à fase atual de desenvolvimento.

REFERÊNCIAS BILBIOGRÁFICAS

[1] https://www.portugal.gov.pt/gc22/portugal-digital/plano-de-acao-para-a-transicao-digital-pdf.aspx
[2] https://files.dre.pt/1s/2020/04/07800/0000600032.pdf
[3] https://education.ec.europa.eu/pt-pt/focus-topics/digital-education/action-plan
[4] https://side.utad.pt/
[5] https://bulletsolutions.com/
[6] https://bullet-roombooking.utad.pt/
[7] https://www.android.com/intl/en/what-is-android/
[8] https://source.android.com/
[9] https://developer.android.com/
[10] https://developer.android.com/kotlin
[11] https://kotlinlang.org/
[12] https://developer.android.com/develop/ui/compose
[13] https://aws.amazon.com/what-is/restful-api/
[14] https://bullet-api.utad.pt/SWAGGER-DOCUMENTATION/index.html
[15] https://square.github.io/retrofit/
[16] https://developer.android.com/training/data-storage/room
[17] https://developer.android.com/topic/architecture/data-layer/offline-first
[18] https://developer.android.com/training/dependency-injection/hilt-android
[19] https://developer.android.com/topic/architecture/intro
[20] https://developer.android.com/topic/modularization
[21] https://developer.android.com/topic/architecture
[22] https://www.uml.org/what-is-uml.htm