INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Curso Técnico Superior Profissional de Tecnologias Web e Dispositivos Móveis

Programação de Aplicação Desktop

Gestão de Clubes Desportivos – SportsUnity

Elaborado por:

Marisa Malveiro, N.º 25247

Nuno Faria, N.º 25906

Docentes:

Henrique Emanuel Água-Doce

João Paulo Trindade

Luís Garcia

Luís André Rosário

Beja

2024

Índice de conteúdos

Introdução 4
Revisão da aplicação desktop 5
Organização do trabalho e ferramentas 6
Análise do problema 7
Funcionalidades da aplicação 8
Desenho da interface da aplicação 9
Desenho da Base de Dados 11
Programação da lógica da aplicação 14
Testes com a aplicação 17
Conceção da página web da aplicação 18
Conclusão 19

Índice de Ilustrações

Figura 1 - Mapa de Navegação 9
Figura 2 - Protótipos de Baixa e Alta Fidelidade dos Ecrãs 10
Figura 3 - Diagrama Entidade-Relação da Base de Dados 11
Figura 4 - Código da criação da Base de Dados 12
Figura 5 - Código da Entidade Club 12
Figura 6 - Operações DAO do entidade Club 13
Figura 7 - Query 13
Figura 8 - Código da função principal 14
Figura 9 - Código de uma parte do sistema de Login da aplicação 15
Figura 10 - Parte do código que exibe treinos 16
Figura 11 - Página Web 18

Introdução

Dando continuidade ao projeto anterior intitulado "Programação de Aplicação Desktop", que tinha como objetivo a gestão eficiente de clubes desportivos através de um terminal em computador, esta nova fase do projeto centra-se na "Programação de Aplicação Lado do Cliente". A principal evolução desta etapa é a extensão do suporte da aplicação a dispositivos móveis, permitindo um acesso mais conveniente e flexível às funcionalidades do sistema.

Os clubes desportivos desempenham um papel crucial na promoção do desporto, saúde, desenvolvimento de habilidades, espírito de equipa e entretenimento. Gerir um clube com um grande número de atletas e informações pode ser desafiante quando dependemos apenas da memória ou de registos em papel. Por isso, o objetivo central deste projeto continua a ser a facilitação da gestão da informação do clube, agora com a vantagem de uma aplicação acessível em qualquer lugar, a qualquer momento.

A aplicação "Gestão de Clube Desportivo" foi melhorada para permitir aos utilizadores a gestão de dados do clube de maneira ainda mais simples e acessível. Na versão para dispositivos móveis, a aplicação está otimizada para a visualização de informações, não incluindo as opções de administração como criação, edição e remoção de dados dos utilizadores do clube, treinos, escalões e competições, que permanecem exclusivas da versão para computadores.

Os ecrãs da aplicação foram desenvolvidos de acordo com os princípios lecionados na disciplina de Sistemas Interativos, do docente Luís Garcia, e o design foi também orientado pela unidade curricular de Design Gráfico, com o docente António Peleja. Adicionalmente, esta versão inclui um site para divulgação da aplicação, elaborado conforme os conhecimentos adquiridos na disciplina de Tecnologias Web, ministrada pelo docente Luís Rosário.

Os utilizadores podem criar um clube, fazer login com os dados fornecidos pelo administrador ou entrar como convidados, sem a necessidade de dados de login. A programação foi realizada em Kotlin, utilizando o Jetpack Compose para a construção da interface do utilizador.

Esta fase do projeto visa proporcionar uma gestão mais eficiente e acessível das informações do clube, aproveitando as vantagens da mobilidade e garantindo que os dados estejam sempre ao alcance dos utilizadores.

Revisão da aplicação desktop

Antes de começarmos a desenvolver a nova aplicação, fizemos uma revisão onde avaliámos vários pontos da aplicação construída da unidade curricular anterior. Avaliamos o que funcionou bem e o que precisa ser melhorado na forma como trabalhamos juntos, como distribuímos as tarefas, verificamos se foram concluídas, quais ferramentas usamos e quais novas tecnologias poderíamos adotar. Também discutimos outros ajustes necessários para facilitar o processo de desenvolvimento do novo projeto.

Organização do trabalho e ferramentas

Para organizar o trabalho e escolher as ferramentas neste projeto, seguimos o mesmo procedimento bem-sucedido do projeto anterior. Começamos com uma análise detalhada das necessidades e requisitos do novo projeto, olhando para o que foi feito anteriormente para evitar erros e maximizar a eficiência.

Dividimos as tarefas de forma clara e equitativa entre nós, assegurando que cada responsabilidade fosse bem definida e acompanhada ao longo do processo. Isso ajudou a manter um fluxo de trabalho organizado e garantir que todos os aspetos do projeto fossem abordados de maneira abrangente.

Para o desenvolvimento da aplicação, utilizamos o Android Studio, uma plataforma especializada para o desenvolvimento de aplicações Android, também por ser o local onde aprendemos a linguagem.

Além disso, utilizámos o GitHub como plataforma de controlo de versões e colaboração. O GitHub facilitou a gestão do código, permitindo que o nosso grupo trabalhasse simultaneamente nos mesmos ficheiros sem conflitos, além de oferecer um controlo rigoroso sobre alterações e versões do projeto.

Estas decisões estratégicas e o método organizacional adotado foram fundamentais para o sucesso do projeto, garantindo que o desenvolvimento fosse eficiente, colaborativo e alinhado com os objetivos estabelecidos desde o início.

Análise do problema

É importante pesquisar projetos semelhantes porque nos ajuda a aprender com o que já foi feito. Podemos ver o que funcionou bem e o que não funcionou em outras aplicações parecidas. Assim, podemos evitar erros e saber o que os utilizadores esperam, garantindo que a nossa aplicação seja útil e funcione bem para quem a usa. Deixamos abaixo um exemplo, de uma aplicação que encontramos, com o nome "TeamSnap", definindo os seus aspetos positivos e negativos:

Aspetos Positivos:

- Simplicidade: Interface simples e fácil de usar, adequada para todos os níveis de habilidade tecnológica.
- Comunicação Eficiente: Ferramentas de comunicação integradas para facilitar a interação entre atletas, treinadores e encarregados de educação.

Aspetos Negativos:

- Escalabilidade Limitada: Pode não ser adequado para clubes muito grandes com necessidades complexas de gestão.
- Integração Limitada: Algumas limitações na integração com outras plataformas e ferramentas externas.

É crucial caracterizar os utilizadores e descrever os cenários de uso da aplicação porque isso assegura que a experiência seja intuitiva e útil para eles. Ao compreender quem são os utilizadores e como vão interagir com a aplicação em diferentes situações, podemos desenvolvê-la de forma a corresponder às suas necessidades específicas, garantindo assim maior eficácia e facilidade de utilização.

Criámos então alguns personas e cenários, do qual deixamos um exemplo abaixo:

João Silva: 10 anos, é estudante do 5º ano na Escola Frei André da Veiga, em Santiago do Cacém, e atleta de futebol no clube Estrela de Santo André.

O João Silva utiliza a aplicação para ver os dados referentes a ele próprio. Os seus treinos mudaram o horário e ele ainda não os decorou. Para visualizar os treinos o João acedeu à aplicação e faz login na aplicação com os seus dados. Depois carregou na opção de treinos e aí consegue visualizar as horas e locais dos mesmos.

1 Ligar telemóvel;
2 Aceder à aplicação "SportsUnity";
3 Fazer login com a conta de atleta;
4 Carregar em treinos;
5 Visualizar os treinos.

Funcionalidades da aplicação

A aplicação permite aos utilizadores visualizar informações detalhadas sobre os clubes desportivos, como modalidades as disponíveis, equipas, horários de treinos, os atletas, calendário de jogos e os seus resultados. Estas funcionalidades são projetadas para oferecer uma visão completa e atualizada das atividades do clube, facilitando o acompanhamento e participação nas atividades desportivas.

No entanto, não serão incluídas opções administrativas, como a criação de novos clubes ou a edição de dados. Esta decisão é justificada por razões práticas e foco do projeto. Ao concentrar-se exclusivamente na visualização de informações, simplifica-se o desenvolvimento e a manutenção da aplicação. Além disso, isso permite que a aplicação seja mais leve e rápida, adequada para dispositivos móveis, e evita a complexidade adicional associada à gestão de dados administrativos.

Desta forma, a aplicação é direcionada para proporcionar uma experiência de utilização fluida e intuitiva aos utilizadores móveis, sem a necessidade de funcionalidades administrativas que poderiam complicar o uso diário da aplicação desportiva.

Desenho da interface da aplicação

Para garantir que a aplicação atende às necessidades dos utilizadores, identificamos as principais tarefas que eles quererão realizar. A aplicação foca-se na visualização de informações sobre clubes desportivos, modalidades, equipas, horários de treino, atletas, calendário de jogos e resultados. As tarefas identificadas são as seguintes:

- Visualizar as modalidades desportivas disponíveis;
- Consultar as equipas de cada modalidade;
- Verificar os horários de treino;
- Obter informações sobre os atletas;
- Aceder ao calendário de jogos;
- Consultar os resultados dos jogos;
- Visualizar informações dos clubes.

Depois de identificar as tarefas dos utilizadores, iniciámos o desenho da interface da aplicação móvel. Este design teve como objetivo garantir uma usabilidade que permitisse aos utilizadores realizar as tarefas de forma intuitiva e eficiente. Antes do desenho, criámos um mapa de navegação, que está na imagem abaixo. Durante o desenvolvimento dos ecrãs, aplicámos os princípios e regras de usabilidade apresentados ao longo do semestre.

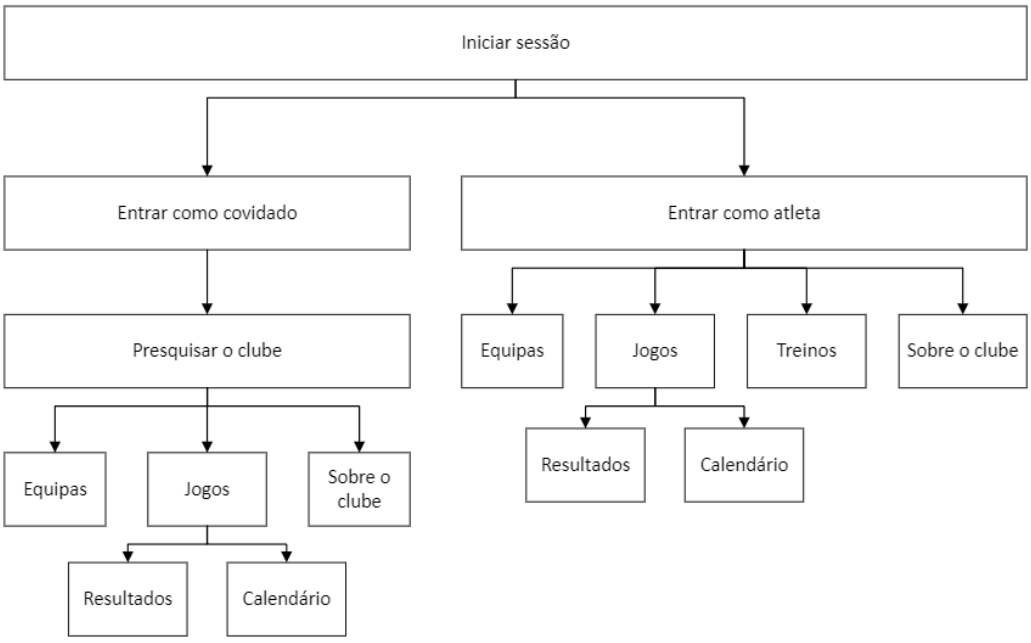

Figura 1 - Mapa de Navegação

A primeira versão dos ecrãs foi criada com lápis e papel para facilitar ajustes rápidos e iterativos. Posteriormente, refinámos esses esboços utilizando uma ferramenta gratuita de prototipagem de ecrãs, permitindo uma visualização mais detalhada e interativa da aplicação.

Ao lado, apresentamos as imagens dos protótipos de baixa e alta fidelidade que desenvolvemos para a aplicação móvel. Estes protótipos ilustram o processo de criação da interface, desde os esboços iniciais em papel até os designs refinados utilizando uma ferramenta gratuita de prototipagem de ecrãs.

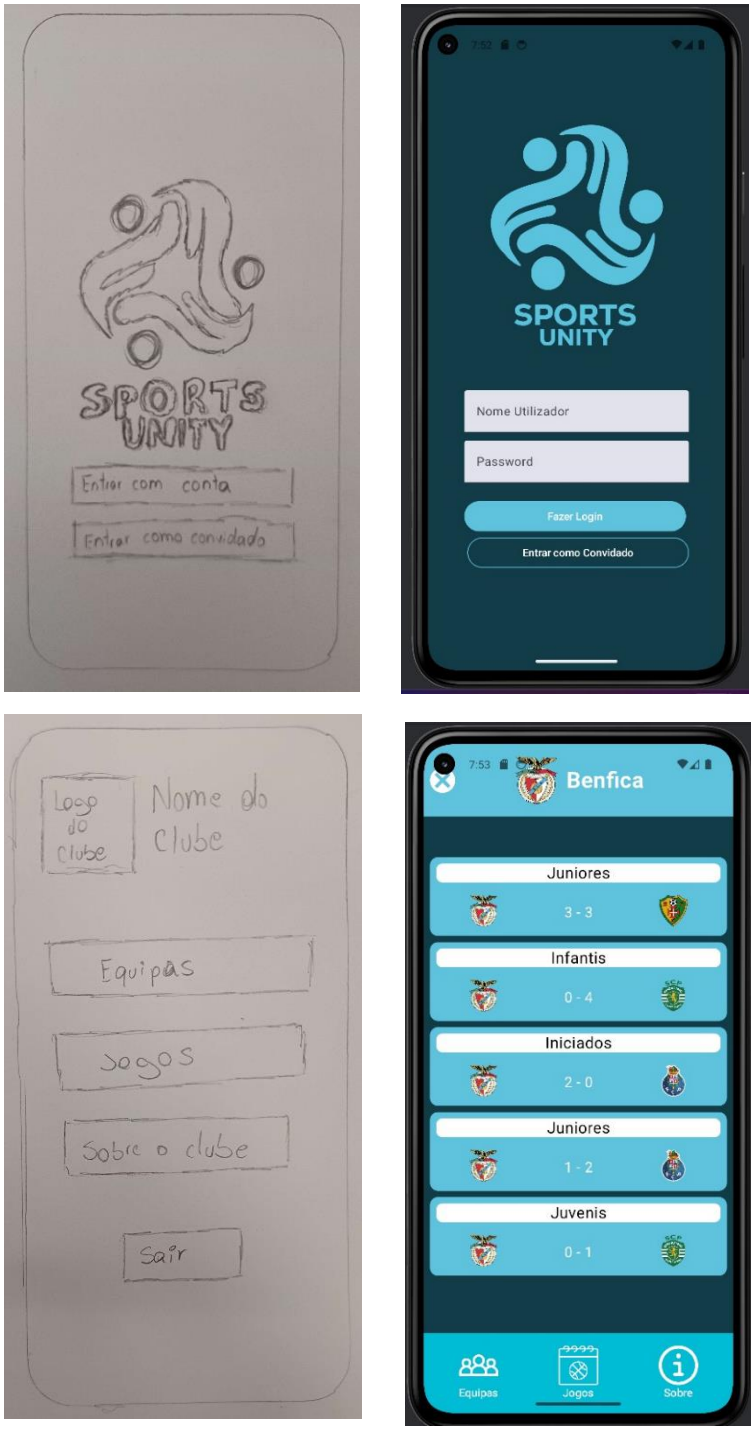

Figura 2 - Protótipos de Baixa e Alta Fidelidade dos Ecrãs

Desenho da Base de Dados

Com base no trabalho desenvolvido nas etapas anteriores e nos conhecimentos adquiridos sobre bases de dados, realizámos uma revisão do desenho conceptual da base de dados. Após uma análise detalhada, concluímos que o desenho da base de dados utilizado no projeto anterior foi bem construído e atende adequadamente às necessidades da aplicação atual. Portanto, não foram necessárias alterações no design conceptual da base de dados.

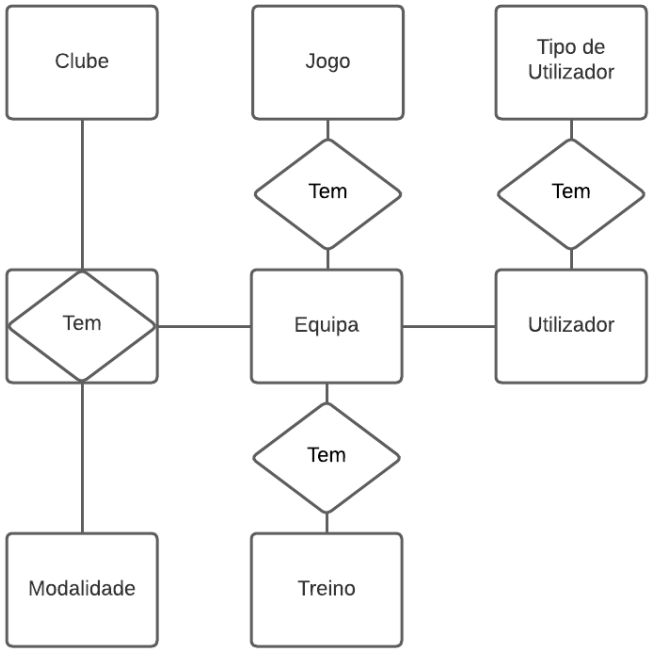

Figura 3 - Diagrama Entidade-Relação da Base de Dados

Conceção da Base de Dados

Partindo já para o Android Studio, iremos descrever a criação da base de dados e algumas queries usadas utilizando a biblioteca Room. Vamos abordar a definição das entidades, os DAOs, as migrações e a configuração geral do banco de dados.

A configuração da base de dados é realizada na classe AppDatabase, que define a estrutura da base de dados.

```
@Database(entities = [User::class, Club::class, Modality::class, ClubModalityCrossRef::class, Game::class, Team::class, Training::class], version = 1, exportSchema = false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
    abstract fun clubDao(): ClubDao
    abstract fun teamDao(): TeamDao
    abstract fun gameDao(): GameDao
    abstract fun trainingDao(): TrainingDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized( lock: this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                     name: "app_database"
                )
                    //.addMigrations(MIGRATION_1_2)
                    .addCallback(DatabaseCallback(context))
                    .build()
                INSTANCE = instance
                instance
            }
        }
```

Figura 4 - Código da criação da Base de Dados

As entidades no Room representam tabelas na base de dados. Cada entidade é uma classe de dados anotada com @Entity. São necessárias para descrever a estrutura das tabelas, como as suas colunas e as chaves primárias.

```
@Entity(tableName = "clubs")
data class Club(
    @PrimaryKey(autoGenerate = true) val id: Int,
    val name: String,
    val address: String,
    val telephoneNum: String,
    val email: String
)
```

Figura 5 - Código da Entidade Club

As operações são definidas através de DAOs (Data Access Objects), abaixo está um exemplo de como são feitas as operações na entidade Club:

```
@Dao
interface ClubDao {
    @Insert
    suspend fun insert(club: Club)

    @Insert
    suspend fun insertAll(clubs: List<Club>)

    @Query("SELECT * FROM clubs")
    fun getAllClubs(): List<Club>

    @Query("SELECT * FROM clubs WHERE id = :id")
    suspend fun getById(id: Int): Club?
}
```

Figura 6 - Operações DAO do entidade Club

As Queries são as funções que fazem pedidos à base de dados. Abaixo está um exemplo, que retorna um registo da tabela clubs com o ID especificado.

```
@Query("SELECT * FROM clubs WHERE id = :id")
suspend fun getById(id: Int): Club?
```

Figura 7 - Query

Programação da lógica da aplicação

Depois de perceber como foi aplicada a base de dados na aplicação, iremos mostrar a seguir as principais funções que fazem tudo funcionar.

Função Principal

A estrutura abaixo define a lógica de navegação da aplicação, onde a MainActivity configura a aplicação e a função MyApp define a estrutura de composição da interface com base na navegação entre diferentes ecrãs (Screens). Cada ecrã recebe dados através de args.club e args.user, permitindo a personalização e exibição dinâmica de conteúdo conforme necessário.

```
@Composable
fun MyApp(dataViewModel: DataViewModel/*, clubViewModel: ClubViewModel*/) {
    val navController = rememberNavController()
    NavHost(
        navController = navController,
        startDestination = LoginScreen
    ){
        composable<LoginScreen> {
            LoginScreenComp(navController = navController,dataViewModel)
        }
        composable<SearchClubScreen> {
            val args = it.toRoute<SearchClubScreen>()
            SearchClubScreenComp(navController = navController, dataViewModel/*, args.teamId*/)
        }
        composable<TeamsScreen> {
            val args = it.toRoute<TeamsScreen>()
            TeamsScreenComp(dataViewModel, navController = navController, args.teamId, args.clubId, args.clubName,args.clubPhoneNum, args.clubEmail, args.clubAddress,args.user)
        }
        composable<GamesResultsScreen> {
            val args = it.toRoute<GamesResultsScreen>()
            GamesResultsScreenComp(dataViewModel, navController = navController, args.teamId, args.clubId, args.clubName,args.clubPhoneNum, args.clubEmail, args.clubAddress,args.user)
        }
        composable<AboutScreen> {
            val args = it.toRoute<AboutScreen>()
            AboutScreenComp(dataViewModel, navController = navController,args.teamId, args.clubId, args.clubName,args.clubPhoneNum, args.clubEmail, args.clubAddress,args.user)
        }
        composable<TrainingScreen> {
            val args = it.toRoute<TrainingScreen>()
            TrainingScreenComp(dataViewModel, navController = navController, args.teamId, args.clubId, args.clubName,args.clubPhoneNum, args.clubEmail, args.clubAddress)
        }
    }
}
```

Figura 8 - Código da função principal

Ecrãs

Cada ecrã tem um ficheiro separado, neste relatório iremos explicar dois:

Ecrã de Login:

Este ecrã tem uma tela com campos de texto para inserção dos dados de login ao carregar no botão "Fazer Login". Ao fazer isso, ele chama o método `viewModel.authenticate(username, password)` que, por sua vez, consulta a base de dados para verificar se o nome de utilizador e a senha correspondem a um utilizador válido. Se a autenticação for bem-sucedida, ele navega para o ecrã de Treinos (`TrainingScreen`). Caso contrário, mostra uma mensagem de erro indicando que os dados de login são inválidos. Este ecrã também tem um botão de “Entrar como Convidado” que redireciona automaticamente para o ecrã de escolher um clube.

```
Button(
    onClick = {
        viewModel.authenticate(username, password) { user, club ->
            if (user != null) {
                if (club != null)
                {
                    navController.navigate(TrainingScreen(user.clubId, club.id, club.name, club.telephoneNum, club.email, club.address))
                }
                else{
                    errorMessage = "Problema ao entrar, tente novamente."
                }
            } else {
                errorMessage = "Nome de utilizador ou senha incorretos."
            }
        }
    },
    colors = ButtonDefaults.buttonColors(containerColor = Color( color: 0xFF5FC3DD)),
    modifier = Modifier
        .weight(1f)
        .padding(end = 4.dp)
) {
    Text( text: "Fazer Login", color = Color.White)
}
```

Figura 9 - Código de uma parte do sistema de Login da aplicação

Ecrã dos Treinos:

Este código, que mostra os horários e local de treinos de um atleta numa equipa, está feito com duas funções:

- TrainingScreenComp: Esta função cria o ecrã da aplicação. Recebe um controlador de navegação (NavHostController) e um modelo de dados (DataViewModel) para gerenciar os dados dos treinos. O ecrã é estruturado em uma coluna com cabeçalho, exibição dos treinos e uma barra de navegação inferior.

- Training: Esta função composta renderiza as informações dos treinos de uma equipa em específico, no caso, a que o atleta tiver guardada nos seus dados. Utiliza estados (remember) para armazenar e atualizar os dados de treino, e usa LaunchedEffect para carregar esses dados. Cada treino é exibido em um Card dentro de uma coluna.

```
LazyColumn {
    items(training.size) { index ->
        Row(modifier = Modifier
            .fillMaxWidth()
            .padding(8.dp)
            .background(Color.White, shape = RoundedCornerShape(8.dp)),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Box(
                contentAlignment = Alignment.Center,
                modifier = Modifier.weight(1f)
            ) {
                Text(text = training[index].dayOfTheWeek, fontSize = 24.sp, color = Color.Black, textAlign = TextAlign.Center)
            }
        }
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Box(
                modifier = Modifier
                    .weight(1f)
                    .padding(horizontal = 14.dp)
            ) {
                Text(text = "Horário:" + training[index].time, fontSize = 20.sp, color = Color.Black, textAlign = TextAlign.Center)
            }
```

Figura 10 - Parte do código que exibe treinos

Testes com a aplicação

Fazer testes na aplicação fazendo avaliações com utilizadores reais é uma etapa crucial no processo de desenvolvimento de aplicações e sistemas. Para garantir a eficácia e a usabilidade do produto final, é essencial envolver utilizadores reais no teste de funcionalidades e na identificação de possíveis melhorias.

Deixamos em baixo duas das avaliações que fizemos para o desenvolvimento desta aplicação.

Avaliação 1
Atleta a consultar horários de treinos
Utilizador - Sofia Palma

Número de cliques: 5
Tempo de conclusão: 15 segundos
Erros cometidos: 0

O utilizador tem um elevado nível de experiência com tecnologia. Conseguiu completar a tarefa com êxito, achando-a extremamente fácil e sem encontrar problemas. A interface foi considerada extremamente intuitiva. Em termos de feedback geral, o utilizador achou a interface visualmente agradável e sentiu-se seguro ao usar o protótipo. Não foram sugeridas melhorias.

Avaliação 2
Atleta a visualizar jogos do seu clube
Utilizador – Nazaré Cavaco

Número de cliques: 6
Tempo de conclusão: 22 segundos
Erros cometidos: 0

O utilizador tem uma experiência elevada com tecnologia, completou a tarefa sem dificuldades e sem encontrar problemas. Considerou a interface extremamente intuitiva e visualmente atraente, e sentiu-se seguro ao usar o protótipo. Não foram indicadas melhorias específicas.

Conceção da página web da aplicação

Ter um site para complementar a aplicação é fundamental porque funciona como uma montra online. É onde podemos apresentar claramente o que a aplicação faz e como pode beneficiar os utilizadores. Além de ser um ponto de contacto para suporte e feedback, o site aumenta a visibilidade da app, tornando-a mais acessível e confiável para um público mais vasto.

A partir das disciplinas de Tecnologias Web e Design Gráfico, criamos um site para a mesma aplicação. Isto torna mais fácil para as pessoas encontrarem a nossa aplicação a partir de qualquer dispositivo conectado à internet. Além disso, o design gráfico ajuda a garantir que a página seja visualmente atraente e fácil de usar.

Figura 11 - Página Web

Conclusão

Para concluir, este projeto de "Programação de Aplicação Lado do Cliente" representa uma etapa significativa na nossa jornada de aprendizagem. Inicialmente centrado numa aplicação desktop por terminal para gestão de clubes desportivos, evoluímos para uma versão móvel mais acessível. Inspirados pelas disciplinas de Sistemas Interativos, Design Gráfico e Tecnologias Web, desenvolvemos ecrãs intuitivos e um site atrativo para promover a aplicação.

Utilizámos tecnologias como Kotlin em Jetpack Compose, aprendendo na prática como desenvolver interfaces intuitivas. Embora este projeto não esteja pronto para utilização real, representa o nosso empenho em aplicar o conhecimento académico para melhorar a gestão de informação nos clubes desportivos.

Esta experiência foi fundamental para consolidar os nossos conhecimentos e habilidades práticas, preparando-nos para desafios futuros na programação e no desenvolvimento de aplicações.