Desenvolvimento para Dispositivos Móveis

Aula 07

Persistência de Dados Local e SQLite

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧠 Objetivos

- Entender o conceito de persistência de dados local.
- Aprender como utilizar o SQLite no Android.
- Criar um banco de dados local para armazenar e recuperar informações.
- Implementar uma aplicação simples com operações CRUD.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Persistência de Dados

🔍 Por que armazenar dados localmente?

Ao desenvolver um aplicativo, muitas vezes é necessário **manter informações disponíveis mesmo após o app ser fechado ou o dispositivo reiniciado**. Isso é o que chamamos de **persistência de dados**.

Exemplos:

- Um app de anotações precisa guardar os textos criados pelo usuário.
- Um app de tarefas precisa lembrar quais tarefas foram concluídas.
- Um app de música pode armazenar as últimas faixas ou playlists acessadas.

UniFAP

🌐 Armazenamento Local vs Remoto

Aspecto	Armazenamento Local	Armazenamento Remoto
Onde os dados são salvos	No dispositivo do usuário	Em um servidor (nuvem)
Acesso a dados	Rápido, sem internet	Requer conexão de rede
Compartilhamento	Apenas no dispositivo	Pode ser sincronizado
Segurança	Menos seguro	Mais controlado (dependendo do back-end)
Exemplos	SQLite, SharedPreferences	Firebase, APIs REST, Supabase

Resumo: Use armazenamento local para **dados simples e temporários** ou quando o app precisa funcionar **offline**. Use remoto quando for necessário **sincronizar dados entre dispositivos** ou **compartilhar informações entre usuários**.

UniFAP

🧰 Opções de Armazenamento Local no Android

1. SharedPreferences

- Armazenamento de **pares chave-valor**, ideal para **configurações simples**.
- Ex: manter o tema escuro ativado, salvar o nome do usuário.
- **Não é indicado para dados complexos** como listas ou tabelas.

```
val prefs = getSharedPreferences("AppPrefs", MODE_PRIVATE)
prefs.edit().putString("username", "João").apply()
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 Opções de Armazenamento Local no Android

2. Arquivos

- Permite **ler e escrever arquivos diretamente**.
- Pode ser usado para **JSON, CSV, ou textos personalizados**.
- Flexível, mas exige mais cuidado com parsing, performance e segurança.

```
openFileOutput("dados.txt", MODE_PRIVATE).use {
    it.write("Texto salvo".toByteArray())
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 Opções de Armazenamento Local no Android

3. SQLite

- Banco de dados **relacional**, embutido no Android.
- Ideal para **estruturas complexas** com várias tabelas e relacionamentos.
- Permite operações **CRUD completas**.
- Requer uso de SQL ou interfaces como SQLiteOpenHelper.

```
val db = SQLiteDatabase.openOrCreateDatabase("meu_banco.db", null)
db.execSQL("CREATE TABLE IF NOT EXISTS Aluno (id INTEGER, nome TEXT)")
```

https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧰 Opções de Armazenamento Local no Android

4. Room

- **Abstração moderna sobre o SQLite**, parte do Jetpack.
- Utiliza **anotações e Data Access Objects (DAOs)**.
- Facilita manutenção e testes.
- **Recomendado para projetos mais robustos**.

vale a pena explorar depois.

```
@Entity
data class Usuario(@PrimaryKey val id: Int, val nome: String)

@Dao
interface UsuarioDao {
    @Query("SELECT * FROM Usuario")
    fun getTodos(): List<Usuario>
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



SQLite

🔎 2. O que é SQLite?

📦 Banco de dados relacional embutido

O **SQLite** é um sistema de **banco de dados relacional** que vem **embutido no Android**, ou seja, **não precisa ser instalado separadamente**. Isso o torna ideal para armazenar dados estruturados diretamente no dispositivo do usuário.

- “Relacional” significa que os dados são organizados em **tabelas**, assim como no MySQL ou PostgreSQL.

- Mas, ao contrário desses sistemas, o SQLite é **leve**, **auto suficiente** e **salva os dados em um único arquivo** no dispositivo.

📌 **Curiosidade:** O nome “SQLite” vem de **SQL (Structured Query Language)** + **Lite (leve)**.

✅ Vantagens do SQLite

Vantagem	Explicação
📱 Integrado ao Android	Já vem pronto para uso, sem precisar de instalação ou configuração extra.
⚡ Rápido e eficiente	Ideal para apps que precisam acessar dados rapidamente e com baixo consumo.
💾 Armazenamento local	Os dados são salvos no próprio dispositivo, ótimo para apps offline.
🗃️ Suporta SQL completo	Você pode usar comandos SQL padrão como `SELECT`, `INSERT`, `UPDATE`, `DELETE`.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

❌ Limitações do SQLite

Limitação	Explicação
🌐 Não sincroniza online	Como é local, os dados não são compartilhados entre dispositivos automaticamente.
🐝 Mais complexo que SharedPreferences	Requer conhecimento básico de SQL e estruturação de tabelas.
🔒 Cuidados com segurança	Os dados podem ser acessados por outras apps se não forem bem protegidos.
💻 Sem suporte a múltiplos usuários concorrentes	Não é feito para situações com muitos acessos simultâneos.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧱 Estrutura básica de um banco SQLite

Você pode pensar em um banco SQLite como uma **planilha** com várias abas:

Tabela (Table)	**Como uma aba da planilha. Ex: Alunos, Produtos, Tarefas**
Colunas	Nome das categorias de dados. Ex: ID, Nome, Curso
Registros	Cada linha da tabela. Ex: 1 - João - Sistemas

📋 Exemplo de tabela Aluno

id	nome	curso
1	João	Sistemas de Info
2	Maria	ADS

UniFAP

🧪 Sintaxe básica de SQL (revisão rápida)

```sql
-- Criar uma tabela
CREATE TABLE Aluno (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT,
    curso TEXT
);
```

```sql
-- Deletar um registro
DELETE FROM Aluno WHERE id = 1;
```

```sql
-- Inserir um dado
INSERT INTO Aluno (nome, curso) VALUES ('João', 'ADS');
```

```sql
-- Consultar dados
SELECT * FROM Aluno;
```

```sql
-- Atualizar dado
UPDATE Aluno SET curso = 'SI' WHERE id = 1;
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🎯 Conclusão

- O SQLite é uma ferramenta poderosa para **persistência local de dados estruturados**.
- Seu uso exige organização de dados em tabelas, mas é **extremamente útil** para criar apps robustos e funcionais **mesmo sem internet**.
- É uma **ponte entre a simplicidade do SharedPreferences e a robustez de uma API externa**.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



🛠️ Prática

Criando um CRUD com SQLite

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Prática: Criando um CRUD com SQLite

Crie um Projeto no Android Studio

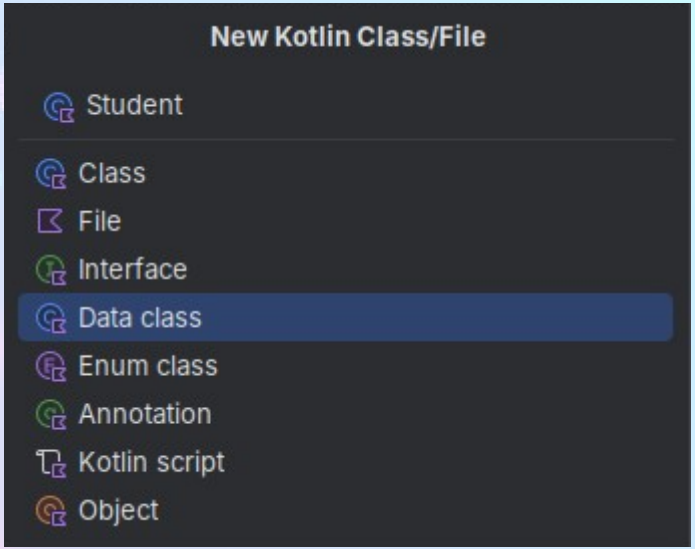
- **Nome**: SQLiteExample
- **Activity**: Empty Views Activity

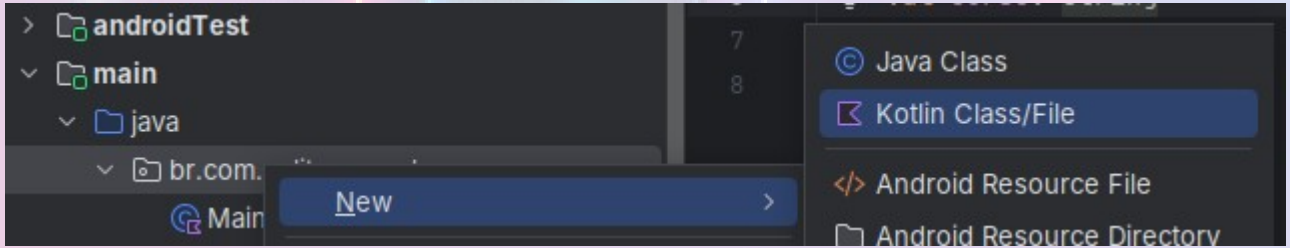
UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Prática: Criando um CRUD com SQLite

Crie uma Classe de Modelo

```
data class Aluno(
    val id: Int,
    val nome: String,
    val curso: String
)
```





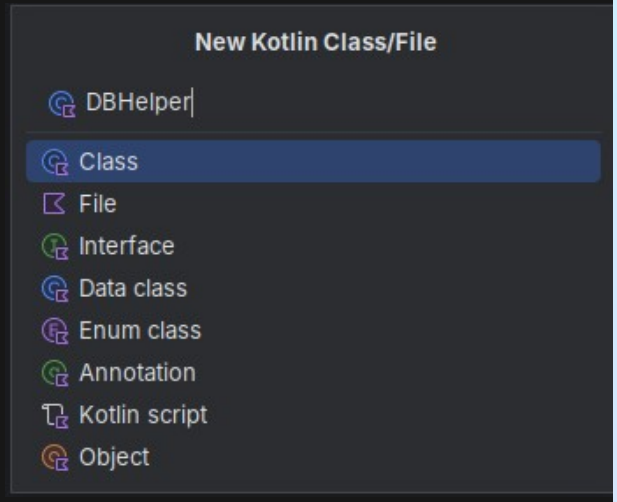
UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🛠️ Prática: Criando um CRUD com SQLite

Criando a Classe do Banco de Dados

```kotlin
class DBHelper(context: Context) : SQLiteOpenHelper(context, "AlunosDB", null, 1) {

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = """
            CREATE TABLE Aluno (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nome TEXT,
                curso TEXT
            );
        """.trimIndent()
        db.execSQL(createTable)
    }
```



🛠️ Prática: Criando um CRUD com SQLite

Criando a Classe do Banco de Dados

```
override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
    db.execSQL("DROP TABLE IF EXISTS Aluno")
    onCreate(db)
}
```

🛠️ Prática: Criando um CRUD com SQLite

Criando a Classe do Banco de Dados

```
fun inserirAluno(nome: String, curso: String): Boolean {
    val db = this.writableDatabase
    val values = ContentValues().apply {
        put("nome", nome)
        put("curso", curso)
    }
    val result = db.insert("Aluno", null, values)
    return result != -1L
}
```

🛠️ Prática: Criando um CRUD com SQLite

Criando a Classe do Banco de Dados

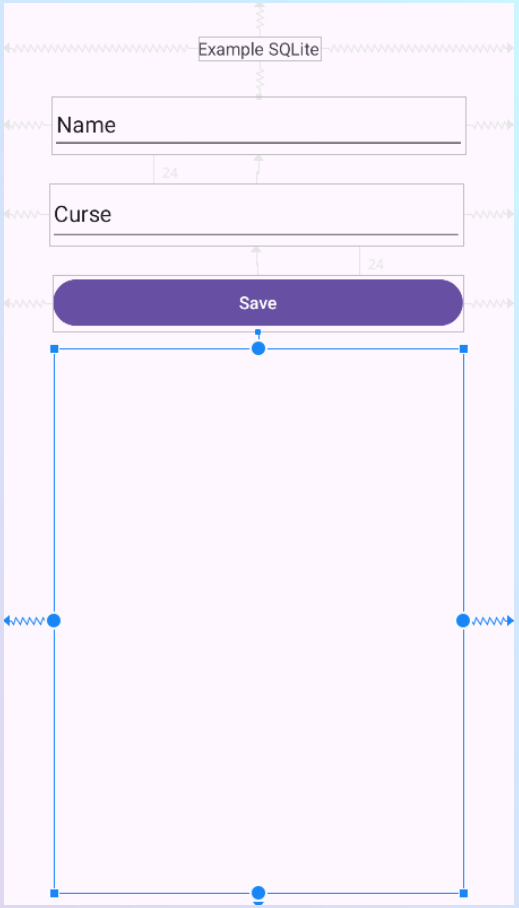
```
fun listarAlunos(): List<Aluno> {
    val alunos = mutableListOf<Aluno>()
    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT * FROM Aluno", null)
    if (cursor.moveToFirst()) {
        do {
            val aluno = Aluno(
                id = cursor.getInt(0),
                nome = cursor.getString(1),
                curso = cursor.getString(2)
            )
            alunos.add(aluno)
        } while (cursor.moveToNext())
    }
    cursor.close()
    return alunos
}
```

🛠️ Prática: Criando um CRUD com SQLite

Criando a Interface do Usuário

activity_main.xml com:

- EditText para nome
- EditText para curso
- Botão para salvar
- TextView (ou RecyclerView) para listar



🛠️ Prática: Criando um CRUD com SQLite

Integração com a MainActivity

```
class MainActivity : AppCompatActivity() {
    private lateinit var dbHelper: DBHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        dbHelper = DBHelper(this)

        val btnSalvar = findViewById<Button>(R.id.btnSalvar)
        val edtNome = findViewById<EditText>(R.id.edtNome)
        val edtCurso = findViewById<EditText>(R.id.edtCurso)
        val txtResultado = findViewById<TextView>(R.id.txtResultado)
```

🛠️ Prática: Criando um CRUD com SQLite

Integração com a MainActivity

```
        btnSalvar.setOnClickListener {
            val nome = edtNome.text.toString()
            val curso = edtCurso.text.toString()
            if (dbHelper.inserirAluno(nome, curso)) {
                Toast.makeText(this, "Aluno inserido!", Toast.LENGTH_SH
                edtNome.text.clear()
                edtCurso.text.clear()
                atualizarLista(txtResultado)
            }
        }

        atualizarLista(txtResultado)
    }
```

🛠️ Prática: Criando um CRUD com SQLite

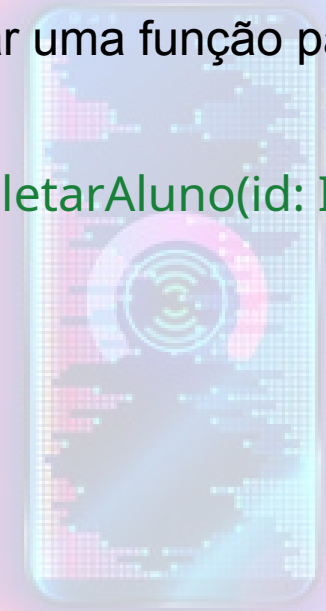
Integração com a MainActivity

```
private fun atualizarLista(txtResultado: TextView) {
    val alunos = dbHelper.listarAlunos()
    txtResultado.text = alunos.joinToString("\n") { "${it.id} - ${it.nome} (${it.curso})" }
}
```

🧪 Exercício

Desafio: Adicionar uma função para deletar um aluno pelo ID.

Dica: criar `fun deletarAluno(id: Int): Boolean` no `DBHelper`.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

🧪 Desafio

🔐 1. SharedPreferences – Salvar Configurações de Usuário

📌 **Objetivo**: Criar um app que permita ao usuário salvar seu **nome** e **tema preferido (claro/escuro)**, e que recupere essas informações ao abrir novamente o app.

🧱 Componentes:

- EditText para o nome
- Switch ou CheckBox para o tema escuro
- Button para salvar
- TextView para mostrar as preferências salvas

💡 Dica de uso:

```
val prefs = getSharedPreferences("UserPrefs", MODE_PRIVATE)
prefs.edit().putString("nome", "Maria").putBoolean("darkMode", true).apply()
```

🧪 Desafio

📂 2. Arquivos – Salvar dados em JSON, CSV e TXT

A) JSON – Lista de Tarefas

📌 **Objetivo**: Criar um app simples de tarefas (ToDo), onde o usuário possa adicionar tarefas a uma lista. A lista será salva como um arquivo JSON no armazenamento interno.

- Exemplo de objeto:

```
[
  { "id": 1, "titulo": "Estudar Kotlin", "concluida": false },
  { "id": 2, "titulo": "Ler Eclesiastes 3", "concluida": true }
]
```

💡 Dica: Use a biblioteca Gson para serializar/deserializar.

```
val json = Gson().toJson(listaDeTarefas)
openFileOutput("tarefas.json", MODE_PRIVATE).use {
    it.write(json.toByteArray())
}
```

🧪 Desafio

📂 2. Arquivos – Salvar dados em JSON, CSV e TXT

B) CSV – Lista de Contatos

📌 **Objetivo**: Criar uma lista simples de contatos com nome e telefone, e salvar os dados em formato .csv.

```
nome,telefone
João,99123-4567
Ana,99345-6789
```

💡 Dica: Gere o conteúdo com StringBuilder e grave com openFileOutput.

🧪 Desafio

📂 2. Arquivos – Salvar dados em JSON, CSV e TXT

C) TXT – Bloco de Notas

📌 **Objetivo**: Criar um pequeno editor de texto onde o usuário possa digitar livremente e salvar o conteúdo em um arquivo .txt.

💡 Dica de gravação e leitura:

```
// salvar
openFileOutput("anotacao.txt", MODE_PRIVATE).use {
    it.write(textoDigitado.toByteArray())
}

// ler
val texto = openFileInput("anotacao.txt").bufferedReader().readText()
```

🧪 Desafio

🧱 3. Room – CRUD de Alunos

📌 **Objetivo**: Criar um app com cadastro completo de alunos, usando Room para persistência.

🧩 **Estrutura:**

🧱 **Componentes:**

- EditText para nome e curso
- Button para adicionar aluno
- RecyclerView para exibir alunos
- Button (ou Swipe) para deletar aluno

💡 Use Room.databaseBuilder(...) para criar a instância do banco.

```
@Entity
data class Aluno(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val nome: String,
    val curso: String
)

@Dao
interface AlunoDao {
    @Insert fun inserir(aluno: Aluno)
    @Query("SELECT * FROM Aluno") fun listar(): List<Aluno>
    @Delete fun deletar(aluno: Aluno)
}

@Database(entities = [Aluno::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun alunoDao(): AlunoDao
}
```

Github

https://github.com/Mobile-Device-Development/SQLiteExample



Bibliografia

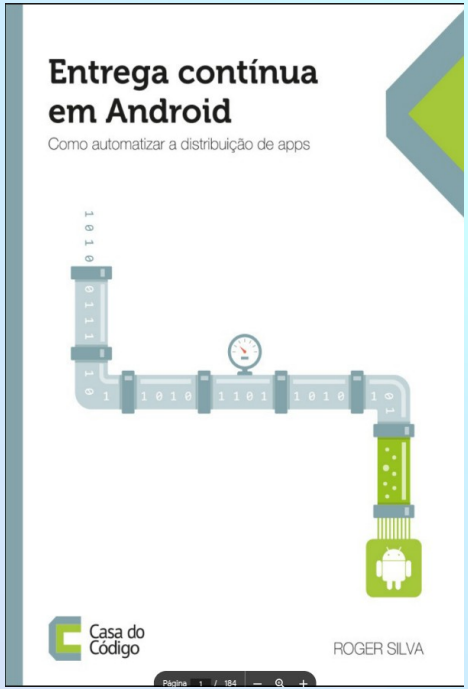
RESENDE, Kassiano. Kotlin com android: crie aplicativos de maneira fácil e divertida. São Paulo, SP: Casa do Código, 2018. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 26 fev. 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/213016

SILVA, Roger. Entrega contínua em android: como automatizar a distribuição de apps. São Paulo, SP: Casa do Código, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 26 fev. 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/212670

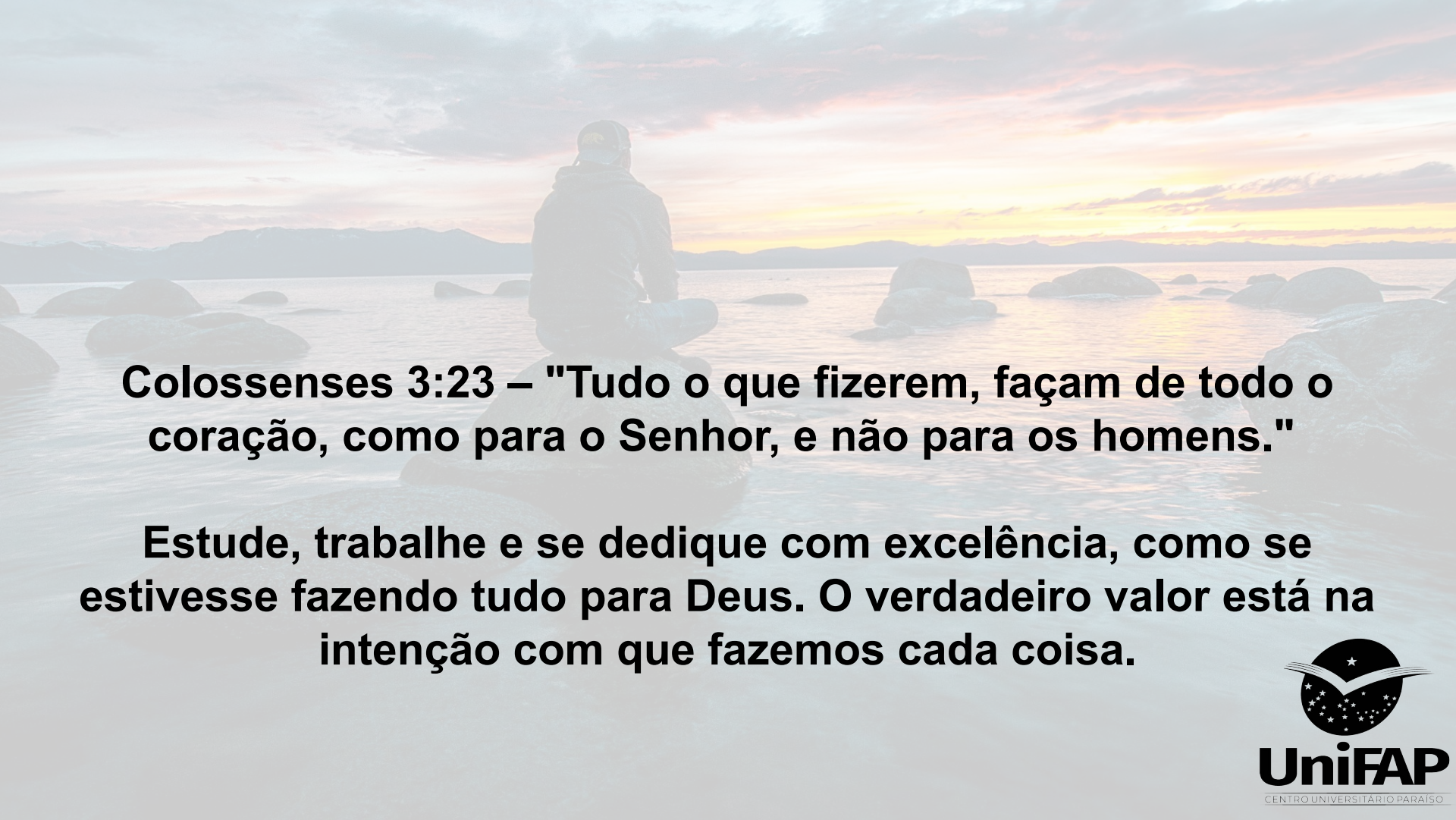




Perguntas?!



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



Colossenses 3:23 – "Tudo o que fizerem, façam de todo o coração, como para o Senhor, e não para os homens."

Estude, trabalhe e se dedique com excelência, como se estivesse fazendo tudo para Deus. O verdadeiro valor está na intenção com que fazemos cada coisa.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



Obrigado!

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO