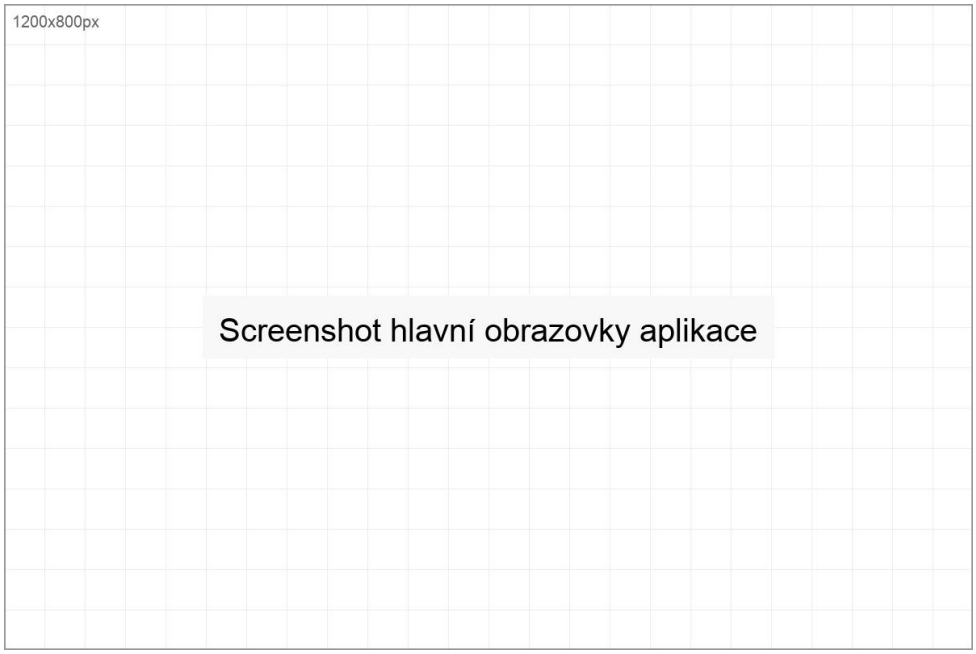
SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Repetito - Mobilní aplikace pro efektivní učení metodou spaced repetition



Autor: Daniel Holeš
Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování
Třída: IT4
Školní rok: 2024/25

Obsah

Poděkování

Děkuji panu učiteli Mgr. Markovi Lucnemu a Ing. Petru Grussmanovi za rady při vytváření tohoto projektu.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2025

. . . . . .

Podpis autora

Abstrakt

Mobilní aplikace Repetito je vzdělávací nástroj využívající metodu spaced repetition pro efektivní učení a zapamatování informací. Aplikace je implementována pomocí frameworku Flutter s využitím Riverpod pro správu stavu a Supabase jako backend řešení. Dokumentace popisuje architekturu aplikace, implementační detaily a způsoby řešení. Důraz je kladen na offline funkcionalitu a synchronizaci dat mezi zařízeními.

Klíčová slova

Flutter, Riverpod, Supabase, spaced repetition, mobilní aplikace, vzdělávání

Abstract

The Repetito mobile application is an educational tool that uses the spaced repetition method for effective learning and information retention. The application is implemented using the Flutter framework with Riverpod for state management and Supabase as a backend solution. The documentation describes the application architecture, implementation details and solutions. Emphasis is placed on offline functionality and data synchronization between devices.

Keywords

Flutter, Riverpod, Supabase, spaced repetition, mobile application, education

SEZNAM POUŽITÝCH ZKRATEK

API	Application Programming Interface,
CRUD	Create, Read, Update, Delete,
HTTP	Hypertext Transfer Protocol,
JSON	JavaScript Object Notation,
REST	Representational State Transfer,
SQL	Structured Query Language,
UI	User Interface,
UX	User Experience.

ÚVOD

Představení projektu

V současné době, kdy se mobilní aplikace staly nedílnou součástí našeho života, roste i potřeba efektivních nástrojů pro vzdělávání. Tradiční metody učení často nevedou k dlouhodobému zapamatování informací, zatímco vědecky podložený přístup spaced repetition využívá přirozených mechanismů lidské paměti. Projekt Repetito vznikl s cílem vytvořit moderní vzdělávací aplikaci, která kombinuje tento osvědčený princip učení s možnostmi současných technologií. Aplikace řeší běžné nedostatky existujících řešení, jako je absence efektivních metod pro dlouhodobé zapamatování a omezená možnost práce offline.

Motivace

Hlavní motivací pro vytvoření aplikace Repetito byla osobní zkušenost s nedostatky současných vzdělávacích aplikací. Většina dostupných řešení buď nenabízí efektivní metody pro dlouhodobé zapamatování, nebo postrádá možnost práce v offline režimu. Současně roste potřeba efektivního učení v době, kdy množství informací, které potřebujeme vstřebat, neustále narůstá. Aplikace Repetito vznikla s cílem poskytnout uživatelům nástroj, který nejen implementuje vědecky podložené metody učení, ale také nabízí moderní a intuitivní uživatelské rozhraní spolu s pokročilými funkcemi jako je synchronizace mezi zařízeními.

Cíle projektu

Hlavním cílem této práce je vytvořit mobilní aplikaci pro efektivní učení pomocí metody spaced repetition. Aplikace musí umožňovat vytváření a správu učebních sad, implementovat algoritmus spaced repetition pro optimální plánování opakování a poskytovat plnohodnotnou offline funkcionalitu. Důležitým aspektem je také synchronizace dat mezi zařízeními a intuitivní uživatelské rozhraní, které uživatelům usnadní pravidelné učení.

Struktura práce

Práce je rozdělena do několika hlavních částí. První kapitola se zabývá teoretickou ástí, která popisuje principy učení, paměti a metodu spaced repetition. Druhá kapitola popisuje frontend aplikace včetně použitého frameworku Flutter a architektury aplikace. Třetí kapitola se věnuje využitým technologiím jako Riverpod a Supabase. Čtvrtá kapitola detailně rozebírá způsoby řešení a použité postupy při implementaci. Pátá kapitola shrnuje dosažené výsledky a práci uzavírá závěrečné zhodnocení projektu s výhledem do budoucna.

1 TEORETICKÁ ČÁST

1.1 Principy učení a paměti

Lidská paměť funguje na základě komplexních neurologických procesů. Při učení dochází k vytváření a posilování synaptických spojení v mozku. Efektivita zapamatování je ovlivněna mnoha faktory, včetně časování opakování, kontextu učení a emocionálního stavu. Výzkumy ukazují, že pravidelné opakování v optimálních intervalech výrazně zvyšuje dlouhodobou retenci informací.

1.2 Metoda Spaced Repetition

Spaced repetition (učení s rozestupy) je technika učení založená na opakování informací v optimálních časových intervalech. Tato metoda vychází z poznatků kognitivní psychologie a výzkumů lidské paměti. Základním principem je, že čím lépe si pamatujeme určitou informaci, tím delí může být interval do dalšího opakování.

1.2.1 Historie a výzkum

Historie metody spaced repetition sahá do konce 19. stolet, kdy německý psycholog Hermann Ebbinghaus provedl prokopnický experimenty v oblasti lidské paměti. Jeho výzkum, publikovaný v roce 1885 v díle "Über das Gedächtnis"(O paměti), položil základy pro pochopení procesu zapomínání a učení.

Ebbinghausovy experimenty byly unikátní svou metodologií:

- Použití nesmyslných slabik pro eliminaci vlivu předchozích znalostí.
- Systematické měření schopnosti vybavit si informace v různých časových intervalech.
- Kvantifikace procesu zapomínání pomocí matematických funkcí.

Na Ebbinghausovu práci navázali v průběhu 20. století další vzkumníci:

- Cecil Alec Mace (1932) - první systematický výzkum efektivity rozloženého učení.
- Paul Pimsleur (1967) - vytvoření exponenciálního systému intervalů pro jazykové učení.
- Sebastian Leitner (1970) - vývoj systému učebních kartiček s různými intervaly opakování.

1.2.2 Ebbinghausova křivka zapomínání

Ebbinghausova křivka zapomínání je matematický model popisující, jak rychle zapomínáme nově naučené informace. Křivka ukazuje, že:

- Největší ztráta informací nastává v prvních hodinách po učen.
- Po 24 hodinách si pamatujeme pouze 40% naučeného.
- Po týdnu klesá retence na přibližně 20%.
- Opakování v správných intervalech výrazně zpomaluje proces zapomínání.

Matematicky lze křivku zapomínání vyjádřit vztahem:

$$R = e^{-t/S} \tag{1.1}$$

kde:

- R je retence (míra zapamatování)
- t je čas od učení
- S je síla paměti

1.3 Analýza existujících řešení

V současné době existuje několik populárních aplikací pro spaced repetition učení, každá s vlastními specifickými přístupy a funkcionalitami:

- Anki
  - Nejrozšířenější open-source řešení.
  - Robustní synchronizační systém.
  - Komplexní možnosti přizpůsobení.
  - Složitější uživatelské rozhraní.
  - Omezená podpora mobilních zařízení.
- Quizlet
  - Populární mezi studenty.
  - Intuitivní uživatelské rozhraní.
  - Rozsáhlá komunita sdílejících obsah.

  - Omezené možnosti spaced repetition.
  - Vyžaduje prmiové předplatné pro pokročilé funkce.

- Memrise
  - Zaměření především na jazykové učení.
  - Propracovaný gamifikační systém.
  - Multimediální obsah.
  - Omezená možnost vlastního obsahu.
  - Závislost na internetovém připojení.

1.3.1 Srovnání funkcionalit

Pro lepší přehled o možnostech jednotlivých aplikací byla vytvořena srovnávací tabulka klíčových funkcionalit:

Funkcionalita	**Anki**	**Quizlet**	**Memrise**
Offline režim	Ano	Částečně	Ne
Synchronizace	Ano	Ano	Ano
Vlastní obsah	Ano	Ano	Omezený
Sdílení obsahu	Ano	Ano	Ne
Statistiky učení	Pokročilé	Základní	Základní
Gamifikace	Ne	Částečně	Ano
Mobilní aplikace	Omezená	Ano	Ano
Zdarma	Ano	Částečně	Částečně

Tabulka 1.1: Srovnání funkcionalit existujících řešení

Z tabulky je patrné, že každá aplikace má své silné a slabé stránky. Zatímco Anki vyniká v pokročilých funkcích a statistikách, postrádá moderní uživatelské rozhraní a gamifikační prvky. Quizlet nabízí dobrou rovnováhu mezi funkcionalitou a použitelností, ale mnoho pokročilých funkcí je dostupných pouze v placené verzi. Memrise má propracovaný systém gamifikace, ale je omezen především na jazykové učení a vyžaduje stálé připojení k internetu.

2 POUŽITÉ TECHNOLOGIE

2.1 Flutter & Dart

Základem aplikace je programovací jazyk Dart ve verzi 3.0.

Klíčové vlastnosti Dart:

- Robustní podpora null safety
- Silný typový systém
- AOT kompilace pro produkci
- JIT kompilace pro vývoj
- Pokročilé asynchronní programování

Flutter představuje moderní open-source UI framework od společnosti Google, který umožňuje vývoj nativních aplikací pro různé platformy z jedné codebáze.

Klíčové vlastnosti frameworku:

- Hot Reload pro okamžité zobrazení změn v kódu
- Bohatá sada předpřipravených Material a Cupertino widgetů
- Vysoký výkon díky kompilaci do nativního kódu
- Jednotná codebáze pro všechny podporované platformy
- Deklarativní pístup k tvorbě UI

2.2 Riverpod

Pro správu stavu aplikace jsem zvolil framework Riverpod.

Klíčové funkce Riverpod:

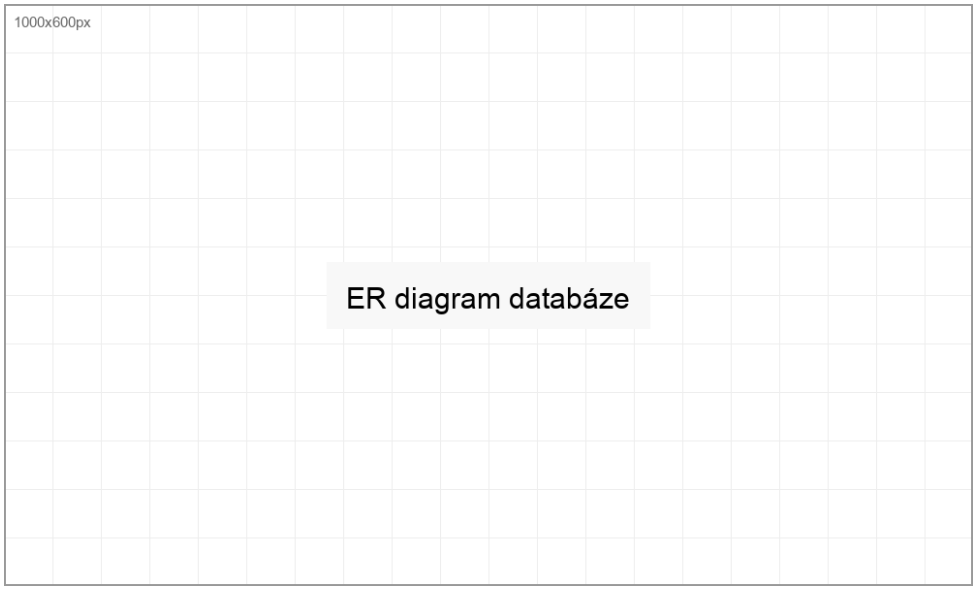
- Typově bezpečné providers
- Integrovaná dependency injection
- Podpora reaktivního programování
- Automatická správa životního cyklu
- Snadná testovatelnost

Implementované typy providerů:

- AsyncNotifierProvider pro asynchronní operace
- NotifierProvider pro synchronní stav
- StreamProvider pro reaktivní datové toky
- StateProvider pro jednoduchý lokální stav
- FamilyProvider pro parametrizované providery

2.3 Supabase

Jako backend řešení jsem implementoval Supabase.



Obrázek 2.1: ER diagram databáze

Klíčové funkce Supabase:

- PostgreSQL databáze
- Realtime subscriptions
- Row Level Security
- REST a GraphQL API
- Vestavná autentizace

2.4 GoRouter

Navigační systém aplikace je postaven na knihovně GoRouter.

Implementované funkce:

- Deklarativní přístup k routingu
- Deep linking podpora
- Nested routes
- Path parametry
- Přehledná správa navigačního stavu

2.5 Freezed

Pro generování kódu využíváme nástroj Freezed.

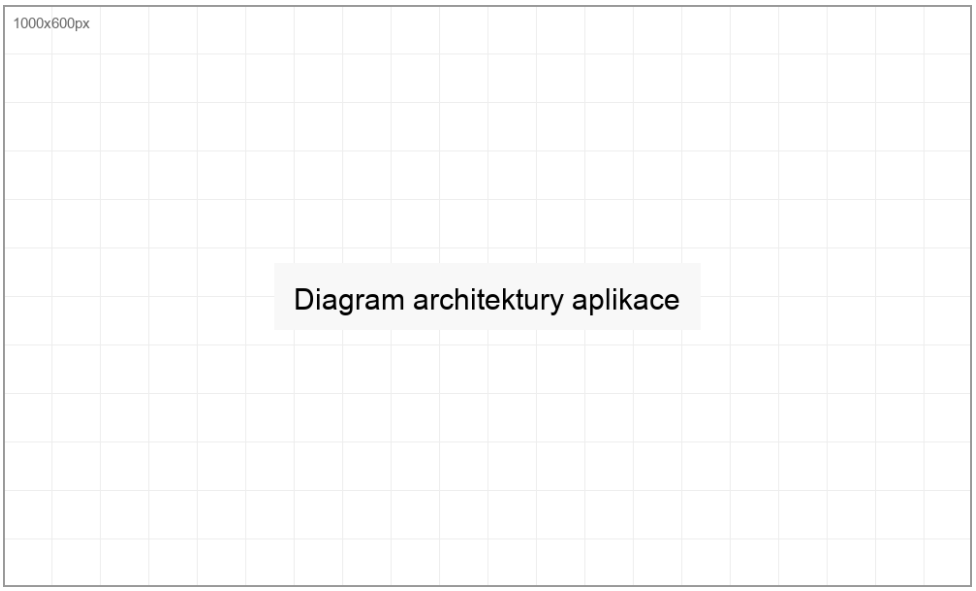
Hlavní přínosy:

- Generování immutable tříd
- Podpora union types
- JSON serializace
- Pattern matching
- Automatické generování equals a hashCode

3 ARCHITEKTURA APLIKACE

3.1 Celková architektura

Aplikaci jsem postavil na principech Clean Architecture, což zajišťuje jasné oddělení business logiky od uživatelského rozhraní.



Obrázek 3.1: Architektura aplikace

Hlavní architektonické principy:

- Lepší testovatelnost jednotlivých komponent
- Snadnější údržba a rozšiřitelnost kódu
- Jasná separace zodpovědností
- Konzistentní terminologie v celém projektu
- Škálovatelnost aplikace

3.2 Datový model

Pro reprezentaci dat v aplikaci jsem použil immutabilní třídy generované pomocí balíčku Freezed. Tento přístup zajišťuje typovou bezpečnost a snadnou serializaci dat.

```
1 // Balíček karet pro učení
2 @freezed
3 class DeckEntity with _$DeckEntity {
4   const factory DeckEntity({
```

```
    required String id,          // Unikátní identifikátor
    required String name,        // Název balíčku
    required String description, // Popis balíčku
    required DateTime createdAt, // Datum vytvoření
    required DateTime updatedAt, // Datum poslední změny
    @Default(false) bool isDeleted,  // Příznak smazání
  }) = _DeckEntity;

  // Deserializace z JSON
  factory DeckEntity.fromJson(Map<String, dynamic> json) =>
      _$DeckEntityFromJson(json);
}

// Jednotlivá učební karta
@freezed
class CardEntity with _$CardEntity {
  const factory CardEntity({
    required String id,          // Unikátní identifikátor
    required String deckId,      // Odkaz na balíček
    required String front,       // Přední strana (otázka)
    required String back,        // Zadní strana (odpověď)
    required DateTime createdAt, // Datum vytvoření
    required DateTime updatedAt, // Datum poslední změny
    @Default(false) bool isDeleted,  // Příznak smazán
  }) = _CardEntity;

  // Deserializace z JSON
  factory CardEntity.fromJson(Map<String, dynamic> json) =>
      _$CardEntityFromJson(json);
}

```

Kód 3.1: Definice datových modelů

4 IMPLEMENTACE

4.1 Frontend implementace

4.1.1 Uživatelské rozhraní

Design uživatelského rozhraní následuje principy Material Design 3. Návrh rozhraní začal vytvořením wireframů klíčových obrazovek:

(a) Wireframe hlavní obrazovky

(b) Wireframe obrazovky studia

Obrázek 4.1: Wireframy klíčových obrazovek

Na základě wireframů byla implementována finální podoba aplikace:

(a) Seznam balíčků

(b) Detail balíčku

Obrázek 4.2: Hlavní obrazovky aplikace

Implementované UI prvky:

- Responzivní layout pro různé velikosti obrazovek
- Adaptivní témata (světlý/tmavý režim)

(a) Obrazovka studia

(b) Statistiky učení

Obrázek 4.3: Studijní a analytické obrazovky

- Plynulé animace a přechody
- Konzistentní typografie a barevné schéma
- Přístupné ovládací prvky

4.2 Backend implementace

Backend aplikace jsem postavil na platformě Supabase, která poskytuje PostgreSQL databázi s realtime funkcionalitou. Implementace zahrnuje:

- Databázové schéma s tabulkami pro balíčky (decks), karty (cards) a záznamy o učení (study_records)
- Row Level Security (RLS) politiky zajišťující přístup uživatelů pouze k vlastním datům
- Realtime synchronizaci pro okamžité aktualizace mezi zařízeními
- Optimalizované databázové dotazy a indexy pro rychlou odezvu

```
-- Politika pro tabulku decks
CREATE POLICY "Uživatelé mohou číst pouze vlastní balíčky"
ON decks FOR SELECT
USING (auth.uid() = user_id);

-- Politika pro tabulku cards
CREATE POLICY "Uživatelé mohou číst karty z vlastních balíčků"
ON cards FOR SELECT
USING (
```

```
  EXISTS (
    SELECT 1 FROM decks
    WHERE decks.id = cards.deck_id
    AND decks.user_id = auth.uid()
  )
);

```

Kód 4.1: Ukázka RLS politik

4.3 Implementace Spaced Repetition

Implementace SuperMemo 2 algoritmu tvoří jádro učebního systému. Algoritmus je implementován v třídě SpacedRepetition, která poskytuje metody pro výpočet intervalů opakování na základě obtížnosti a počtu opakování.

```
class SpacedRepetition {
  // Výpočet příštího intervalu opakování
  static DateTime calculateNextReview(
    DifficultyLevel difficulty,    // Obtížnost odpovědi
    int repetitionCount,           // Počet opakování
    DateTime lastReviewedAt,       // Poslední opakování
  ) {
    // Základní intervaly v hodinách
    final baseInterval = switch (difficulty) {
      DifficultyLevel.easy => 24 * 2.5,    // 2.5 dne
      DifficultyLevel.medium => 24,         // 1 den
      DifficultyLevel.hard => 6,            // 6 hodin
    };

    // Výpočet finálního intervalu
    final multiplier = (1.5 * repetitionCount).clamp(1.0, 30.0);
    final intervalHours = (baseInterval * multiplier).round();
    final randomFactor = 0.9 + (DateTime.now().millisecondsSinceEpoch % 200) / 10
    final finalIntervalHours = (intervalHours * randomFactor).round();

    return lastReviewedAt.add(Duration(hours: finalIntervalHours));
  }
}

```

Kód 4.2: Implementace algoritmu Spaced Repetition

4.4 State Management

Pro správu stavu aplikace využívám Riverpod, který poskytuje typově bezpečný a deklarativní přístup ke správě stavu. Následující ukázka demonstruje implementaci providerů pro správu balíčků a kartiček.

```
@riverpod
class CurrentDeckNotifier extends _$CurrentDeckNotifier {
  @override
  AsyncValue<DeckEntity?> build() => const AsyncValue.data(null);

  Future<void> loadDeck(String deckId) async {
    state = const AsyncValue.loading();
    try {
      final repository = ref.read(deckRepositoryProvider);
      final deck = await repository.getDeck(deckId);
      state = AsyncValue.data(deck);
    } catch (error, stack) {
      state = AsyncValue.error(error, stack);
    }
  }
}

@riverpod
Future<List<CardEntity>> cardList(
  CardListRef ref,
  String deckId,
) async {
  final repository = ref.read(cardRepositoryProvider);
  return repository.getCards(deckId);
}

```

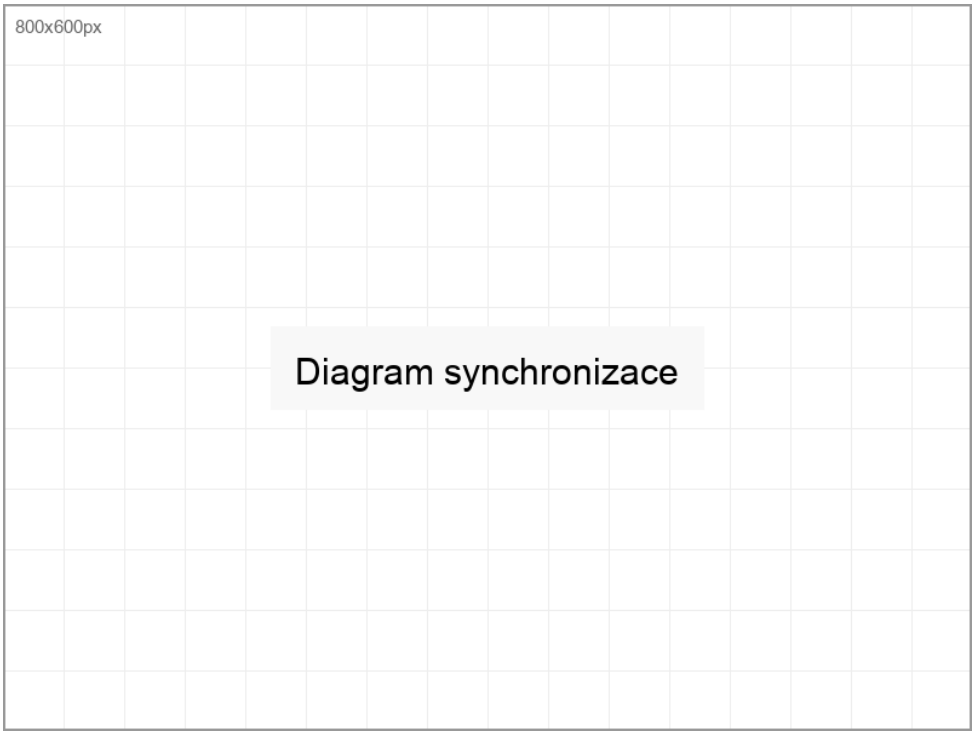
Kód 4.3: Implementace Riverpod providerů

4.5 Offline funkcionalita

Implementace offline režimu je založena na lokální SQLite databázi.

Klíčové aspekty offline funkcionality:

- Plnohodnotná práce bez připojení k internetu
- Automatická synchronizace při obnovení připojení
- Sofistikované řešení konfliktů
- Queue systém pro operace
- Garance konzistence dat



Obrázek 4.4: Diagram synchronizace dat

Implementace offline funkcionality je založena na lokální SQLite databázi a systému front pro synchronizaci změn. Následující ukázka představuje implementaci synchronizačního manageru.

```
class SyncManager {
  final SupabaseClient _client;    // Klient pro Supabase
  final LocalDatabase _localDb;    // Lokální databáze

  SyncManager(this._client, this._localDb);
```

```
6
7   Future<void> synchronize() async {
8     // Kontrola připojení k internetu
9     if (!await _hasInternetConnection()) {
10      debugPrint('Není připojení k internetu, synchronizace přeskočena');
11      return;
12    }
13
14    try {
15      // Proces synchronizace
16      final localChanges = await _getLocalChanges();
17      final serverChanges = await _getServerChanges();
18
19      // ešení konfliktů a aplikace změn
20      final resolvedChanges = await _resolveConflicts(
21        localChanges,
22        serverChanges,
23      );
24      await _applyChangesToServer(resolvedChanges.serverUpdates);
25      await _updateLocalDatabase(resolvedChanges.localUpdates);
26
27      debugPrint('Synchronizace úspěšně dokončena');
28    } catch (e) {
29      debugPrint('Chyba při synchronizaci: $e');
30      rethrow;
31    }
32  }
33 }
34
```

Kód 4.4: Implementace synchronizace

5 VÝSLEDKY A ZHODNOCENÍ

5.1 Splněné cíle

V rámci projektu se mi podařilo úspěšně implementovat všechny klíčové funkcionality a dosáhnout významných výsledků v oblasti mobilního vzdělávání.

Dosažené milníky:

- Implementace algoritmu Spaced Repetition pro efektivní učení
- Realizace robustní offline funkcionality s automatickou synchronizací
- Vytvoření intuitivního uživatelského rozhraní s důrazem na UX
- Optimalizace výkonu aplikace a efektivní využití systémových zdrojů
- Implementace spolehlivého systému synchronizace mezi zařízeními
- Vytvoření komplexního systému pro správu učebních sad
- Zajištění bezpečného ukládání a přenosu dat
- Vytvoření solidního základu pro budoucí rozšíření

5.2 Zhodnocení projektu

Projekt Repetito úspěšně naplnil stanovené cíle v oblasti mobilního vzdělávání a přinesl několik významných inovací.

Hlavní přínosy projektu:

- Moderní technologický stack využívající Flutter, Riverpod a Supabase
- Efektivní implementace Spaced Repetition algoritmu pro optimální učení
- Robustní offline funkcionalita umožňující práci bez připojení k internetu
- Spolehlivá synchronizace dat mezi zařízeními
- Intuitivní uživatelské rozhraní s důrazem na UX

5.3 Budoucí rozvoj

Na základě zkušeností z vývoje a zpětné vazby uživatelů byly identifikovány oblasti pro další rozvoj aplikace.

Plánovaná vylepšení:

- Implementace nových typů učebního obsahu (audio, video)
- Rozšíření statistik učení o pokročilé analytické nástroje
- Optimalizace výkonu a snížení spotřeby baterie
- Přidání gamifikačních prvků pro zvýšení motivace
- Integrace umělé inteligence pro personalizaci učebního procesu
- Rozšíření možností sdílení a spolupráce mezi uživateli

6 ZÁVĚR

Mobilní aplikace Repetito představuje komplexní řešení pro efektivní učení pomocí metody spaced repetition. Projekt jsem rozdělil do tří hlavních částí: frontend aplikace vyvinuté ve frameworku Flutter, backend infrastruktury postavené na Supabase a sofistikovaného algoritmu pro optimalizaci učebního procesu.

V aplikaci jsem úspěšně implementoval všechny klíčové funkcionality včetně správy učebních sad, offline režimu a synchronizace mezi zařízeními. Důraz jsem kladl na vytvoření intuitivního uživatelského rozhraní a optimalizaci výkonu. Využitím moderních technologií jako Riverpod pro správu stavu a Freezed pro generování kódu jsem zajistil robustní a udržitelnou architekturu.

V průběhu vývoje jsem identifikoval možnosti dalšího rozvoje, které by mohly aplikaci posunout na další úroveň. Mezi hlavní plánovaná vylepšení patří implementace pokročilých statistik učení, integrace multimediálního obsahu a přidání gamifikačních prvků. Potenciál aplikace vidím nejen v individuálním vzdělávání, ale také ve školním prostředí nebo firemním vzdělávání.

Zdrojový kód projektu je dostupný na GitHubu (`https://github.com/deathfrost12/repetito`).

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Flutter Team. *Flutter Documentation* [Online]. 2024 [cit. 2024-01-15]. Dostupné z: `https://flutter.dev/docs`

[2] Supabase Team. *Supabase Documentation* [Online]. 2024 [cit. 2024-01-15]. Dostupné z: `https://supabase.com/docs`

[3] Remi Rousselet. *Riverpod Documentation* [Online]. 2024 [cit. 2024-01-15]. Dostupné z: `https://riverpod.dev/docs`

[4] P. Wozniak. *SuperMemo 2: An algorithm for intelligent learning* [Online]. 1998 [cit. 2024-01-15].

[5] Google. *Material Design* [Online]. 2024 [cit. 2024-01-15]. Dostupné z: `https://m3.material.io/`