Analýza a návrh projektu Sense Break NSS

Semestrální projekt

Návrh softwarových systémů (B6B36NSS)

Odkaz na GitLab s podrobným popisem:

gitlab/sense-break-nss

Fakulta elektrotechnická, ČVUT v Praze

Vedoucí: Jiří Šebek
Autorka: Kamilla Ishmukhammedova

Datum: duben 2025

Obsah

Úvod

Tento dokument specifikuje cíle, požadavky a návrh systému **Sense Break**, semestrálního projektu v rámci kurzu **B6B36NSS – Návrh softwarových systémů**.

Sense Break je desktopová aplikace určená především pro vývojáře a uživatele, kteří tráví dlouhý čas u počítače. Poskytuje jednoduchá tréninková cvičení zaměřená na zrak a sluch – například sledování pohybujících se objektů nebo rozpoznávání tónů. Cílem projektu je podpořit prevenci přetížení smyslového vnímání a přispět k lepší péči o zdraví uživatelů formou krátkých interaktivních přestávek.

Aplikace je postavena na mikroservisní architektuře s oddělenými službami pro správu uživatelů, zpracování tréninků a notifikace. Projekt klade důraz na čistý návrh architektury, srozumitelnou dokumentaci, využití návrhových vzorů a použití standardních technologií s důrazem na bezpečnost, modularitu a jednoduchost nasazení.

Motivace projektu

Dlouhodobá práce na počítači představuje značnou zátěž pro zrak i sluch, zejména u vývojářů, grafiků a dalších profesí, které tráví u obrazovky mnoho hodin denně. V důsledku toho může docházet k únavě očí, poklesu koncentrace nebo i chronickým potížím jako je syndrom suchého oka nebo snížená schopnost vnímat vysoké frekvence.

Hlavní motivací projektu **Sense Break** je vytvořit nástroj, který uživatelům pomůže tyto dopady minimalizovat pomocí pravidelných a krátkých tréninků zaměřených na smyslové vnímání. Cvičení budou navržena tak, aby byla snadno přístupná, nenáročná na provedení a vhodná jako součást každodenních „mikropřestávek“ během pracovního dne.

Zároveň je cílem projektu umožnit další rozšiřování funkcí a tréninkových modulů, čímž se aplikace může přizpůsobit různým typům uživatelů i jejich specifickým potřebám. Volba mikroservisní architektury reflektuje snahu o oddělení zodpovědností a snadnou údržbu systému do budoucna.

Cíl zadání

Cílem projektu **Sense Break** je:

1. Navrhnout a realizovat desktopovou aplikaci zaměřenou na trénink zraku a sluchu formou jednoduchých cvičení
2. Rozdělit systém do samostatných mikroslužeb pro správu uživatelů, tréninkovou logiku a notifikace
3. Umožnit uživatelům registraci, přihlášení a sledování výsledků jednotlivých tréninků
4. Navrhnout jednoduché REST API pro komunikaci mezi službami a klientskou aplikací
5. Vytvořit desktopové GUI v technologii JavaFX pro pohodlné ovládání a zobrazení výsledků
6. Zajistit možnost snadného rozšíření o nové typy cvičení nebo služeb do budoucna
7. Připravit dokumentaci systému včetně funkčních a nefunkčních požadavků, UML diagramů a popisu architektury
8. Zohlednit principy použitelnosti, modularity a základní bezpečnosti při návrhu a implementaci

Cílová skupina

Aplikace **Sense Break** je určená zejména pro tyto skupiny uživatelů:

1. Vývojáři, programátoři a IT pracovníci, kteří tráví většinu pracovní doby před obrazovkou počítače
2. Studenti technických oborů, kteří potřebují pravidelné přestávky během dlouhého učení nebo online výuky
3. Pracovníci na home office s omezeným pohybem a vysokou mírou práce s monitorem
4. Uživatelé s oslabeným zrakem nebo sluchem, kteří si chtějí udržovat funkční schopnosti pomocí nenáročných cvičení
5. Firmy a instituce hledající nástroje pro prevenci digitální únavy u svých zaměstnanců

Technologie

Backend

1. **Java 17** – hlavní programovací jazyk pro všechny části backendu
2. **Spring Boot** – framework pro tvorbu REST API a backendových modulů
3. **Spring Web** – obsluha HTTP požadavků a definice REST endpointů
4. **Spring Data JPA (Hibernate)** – ORM pro přístup k databázi
5. **Spring Security (volitelně)** – pro případné zabezpečení přístupu (pokud bude použito)

Desktopová aplikace

1. **JavaFX** – grafické uživatelské rozhraní pro spouštění tréninků a práci s výsledky
2. **HTTP klient (Java 11+)** – komunikace s REST API ze strany desktopového klienta

Databáze

1. **PostgreSQL** – relační databáze pro ukládání uživatelů, tréninkových relací a výsledků

Cache

1. **Redis** – in-memory databáze používaná jako cache vrstva pro urychlení přístupu k často čteným datům (např. uživatelé, historie tréninků, oprávnění).

Messaging

1. **Apache Kafka** – message broker pro asynchronní komunikaci mezi službami
2. **Spring Kafka** – knihovna pro integraci Spring Boot aplikací s Kafka tématy

Verzování a automatizace

1. **GitLab** – verzovací systém a repozitář projektu

Architektura projektu

Typ architektury

Aplikace **SenseBreak** je navržena jako dvousložkový distribuovaný systém, který vychází z principů **modulárního monolitu s oddělenou mikroslužbou**. Architektura je připravena na budoucí rozdělení do plnohodnotných mikroservis.

Systém se skládá ze dvou částí:

1. **Backend Service -** Hlavní služba obsahující většinu aplikační logiky (uživatelé, autentizace, tréninky, výsledky). Je navržena jako modulární monolit – obsahuje oddělené moduly komunikující přes rozhraní (API) a připravené na možnou dekompozici.

2. **Notification Service -** Samostatná mikroslužba zaměřená na asynchronní zpracování událostí (např. dokončení tréninku) a odesílání notifikací uživatelům.

Komunikace mezi částmi systému

- **Frontend – Backend:**
  Probíhá synchronně pomocí **REST API**.

- **Backend – Notification Service:**
  Probíhá **asynchronně pomocí Apache Kafka**, což zajišťuje vysokou škálovatelnost, nezávislost a robustnost komunikace.

Modulární návrh backendu

Backend Service obsahuje tři hlavní moduly:

- **User modul** – správa uživatelů, registrace, přihlášení, profil
- **Training modul** – spuštění tréninku, ukládání výsledků, historie
- **Auth Middleware** – ověření přístupu k chráněným částem API

Každý modul:

- má **vlastní API kontrakt** (rozhraní),
- může být v budoucnu oddělen jako samostatná služba,
- ukládá data do relační databáze **PostgreSQL**, přičemž návrh schématu umožňuje jejich budoucí **datové oddělení**.
- ukládá data do relační databáze **PostgreSQL**, přičemž návrh schématu umožňuje jejich budoucí **datové oddělení**.

- pro urychlení přístupu k často čteným datům je nasazen cache layer **(Redis),** který snižuje zátěž na databázi a zrychluje odpovědi modulů.

Cache vrstva

Pro zajištění rychlejšího přístupu k často využívaným datům (např. uživatelské profily, historie tréninků, oprávnění) je systém doplněn o cache vrstvu pomocí **Redis**.

Při každém požadavku systém nejprve zkontroluje, zda nejsou data dostupná v cache (tzv. **cache hit**). Pokud nejsou, načtou se z databáze a uloží zpět do cache (**cache miss**).

Tímto způsobem se:

- snižuje zatížení relační databáze,
- zrychlují odpovědi backendu,
- zvyšuje celkový výkon systému.

Asynchronní zpracování událostí

Po dokončení tréninku je událost publikována do **Kafka tématu**, kde ji následně zpracuje **Notification Service**. Tento přístup minimalizuje vazby mezi službami a připravuje půdu pro škálovatelnost i výkonnostní optimalizace.

Popis služeb systému

1. Backend Service

Popis: Hlavní služba systému, poskytuje REST API pro desktopového klienta a zahrnuje tři hlavní moduly aplikace: správu uživatelů, autentizaci, tréninky a výsledky. Všechny moduly jsou navrženy jako oddělitelné komponenty s vlastním kontraktem. Často používaná data (např. přihlášený uživatel, uživatelské role, historie tréninků) jsou ukládána do Redis cache, čímž se snižuje zátěž databáze a zrychluje odpověď systému.

Moduly a funkcionality:

User modul

- Registrace nových uživatelů (**POST** `/api/auth/register`)
- Přihlášení uživatelů (**POST** `/api/auth/login`)
- Správa profilu přihlášeného uživatele (**GET/PUT** `/api/users/me`)
  Vybrané profily se mohou načítat z Redis cache

Training modul

- Spuštění nového tréninku (**POST** `/api/trainings/start`)
- Zaznamenání výsledku tréninku (**POST** `/api/trainings/:id/result`)
- Načtení historie tréninků (**GET** `/api/trainings/history`)
  Historie může být čtena z cache
- Získání detailu relace (**GET** `/api/trainings/:id`)

Auth Middleware

- Ověření, že požadavek obsahuje platný identifikátor uživatele
- Odmítnutí přístupu při chybějící nebo neplatné identifikaci (**401 Unauthorized**)
  Ověření může číst role z cache

Po dokončení tréninku backend publikuje zprávu do **Apache Kafka**, na kterou reaguje Notification Service.

Použité technologie: Java 17, Spring Boot, PostgreSQL, Spring Data JPA, Apache Kafka, Redis

2. Notification Service

Popis: Oddělená mikroslužba zpracovávající asynchronní události (např. dokončení tréninku). Naslouchá tématům v Apache Kafka a odesílá notifikace uživatelům (např. připomenutí cvičení).

Funkcionality:

- Příjem zpráv z Kafka témat
- Zpracování události `trénink dokončen`
- Odeslání notifikace klientovi (např. push zpráva)
- Možnost rozšíření o e-mail, systémové oznámení nebo desktop popup

Použité technologie: Java 17, Spring Boot, Apache Kafka, Spring Kafka

Hlavní entity aplikace

1. Uživatel (User)

Popis: Základní entita reprezentující každého uživatele aplikace. Uživatel má přiřazenou roli, která určuje jeho přístup k funkcím systému.

Atributy:

- user_id (UUID) – jednoznačný identifikátor uživatele
- username (String) – zobrazované jméno uživatele
- email (String) – unikátní e-mailová adresa
- password_hash (String) – bezpečně uložené heslo
- role (Enum: user, admin) – uživatelská role
- created_at (Timestamp) – datum registrace
- last_login (Timestamp) – poslední přihlášení

2. Tréninková relace (TrainingSession)

Popis: Entita představující jedno konkrétní spuštěné cvičení (trénink) uživatele.

Atributy:

- session_id (UUID) – jednoznačný identifikátor relace
- user_id (UUID, FK → User) – odkaz na uživatele
- type (Enum: vision, hearing) – typ tréninku
- start_time (Timestamp) – začátek cvičení
- end_time (Timestamp) – konec cvičení
- score (Integer) – výsledek tréninku

3. Výsledek cvičení (TrainingResult)

Popis: Podrobnější výstup z tréninku s metrikami jako přesnost, reakční čas nebo úspěšnost.

Atributy:

- result_id (UUID) – identifikátor výsledku
- session_id (UUID, FK → TrainingSession) – odkaz na trénink
- reaction_time_ms (Integer) – průměrný čas reakce
- accuracy_percent (Float) – procentuální úspěšnost

4. Notifikace (Notification)

Popis: Systémové upozornění uživateli – např. připomenutí, že je čas na další trénink.

Atributy:

- notification_id (UUID) – identifikátor notifikace
- user_id (UUID, FK → User) – cílový uživatel
- type (Enum: reminder, info) – typ notifikace
- message (String) – obsah zprávy
- scheduled_at (Timestamp) – plánovaný čas odeslání
- status (Enum: pending, sent, read) – stav notifikace

Use-Case scénáře

1. Registrace uživatele

Popis: Nový uživatel se registruje pomocí e-mailu, jména a hesla.

Kroky:

1. Uživatel zadá registrační formulář: jméno, e-mail, heslo.
2. Backend Service ověří validitu údajů.
3. Backend vytvoří nového uživatele v databázi.
4. Uživatel je automaticky přihlášen nebo přesměrován na přihlášení.

Výsledek: Uživatel je zaregistrován a připraven používat aplikaci.

Služby: Backend Service (User modul, Auth Middleware)

2. Přihlášení uživatele

Popis: Existující uživatel se přihlašuje pomocí e-mailu a hesla.

Kroky:

1. Uživatel zadá přihlašovací údaje.
2. Backend ověří platnost údajů (porovnání hashe).
3. Uživatel je přihlášen a získá přístupový token.

Výsledek: Uživatel získá přístup ke svému účtu.

Služby: Backend Service (User modul, Auth Middleware)

3. Spuštění tréninku

Popis: Uživatel spustí cvičení zaměřené na zrak nebo sluch.

Kroky:

1. Uživatel si zvolí typ tréninku (zrak / sluch).
2. Backend vytvoří novou relaci tréninku.
3. Desktop klient provede trénink a sleduje výsledek.
4. Výsledek je po dokončení odeslán zpět na server.
5. Backend uloží výsledek do databáze.
6. Backend publikuje událost `trénink dokončen` do Kafka.

7. Notification Service obdrží událost a odešle notifikaci.

Výsledek: Tréninková relace a výsledek jsou uloženy, uživatel je notifikován.

Služby: Backend Service (Training modul), Notification Service, Desktop GUI

Databázové schéma

(Entity-Relationship Diagram)

1. Tabulka: Users

Sloupec	Typ	Popis
user_id	UUID (PK)	Jedinečný identifikátor uživatele
name	String	Jméno uživatele
email	String (unikátní)	E-mailová adresa
password	String	Heslo (plaintext nebo základní hash, podle potřeby)
created_at	Timestamp	Datum registrace
last_login	Timestamp	Datum posledního přihlášení

2. Tabulka: TrainingSessions

Sloupec	Typ	Popis
session_id	UUID (PK)	Jedinečný identifikátor relace
user_id	UUID (FK → Users)	Odkaz na uživatele
type	Enum	Typ cvičení (vision, hearing)
start_time	Timestamp	Čas zahájení tréninku
end_time	Timestamp	Čas ukončení tréninku
created_at	Timestamp	Datum vytvoření relace

3. Tabulka: TrainingResults

Sloupec	Typ	Popis
result_id	UUID (PK)	Identifikátor výsledku
session_id	UUID (FK → TrainingSessions)	Odkaz na tréninkovou relaci
reaction_time	Integer (ms)	Průměrný čas reakce
accuracy	Float (%)	Úspěšnost v procentech
score	Integer	Výsledné skóre
created_at	Timestamp	Datum uložení výsledku

4. Tabulka: Notifications

Sloupec	Typ	Popis
notification_id	UUID (PK)	Identifikátor notifikace
user_id	UUID (FK → Users)	Cílový uživatel
type	Enum	Typ (reminder, info)
message	String	Obsah zprávy
scheduled_at	Timestamp	Plánovaný čas odeslání
status	Enum	Stav (pending, sent, read)

Vztahy mezi tabulkami

1. **User** – má mnoho tréninkových relací (**1:N**, `users.id` → `training_sessions.user_id`)
2. **User** – má mnoho notifikací (**1:N**, `users.id` → `notifications.user_id`)
3. **TrainingSession** – má mnoho výsledků (**1:N**, `training_sessions.id` → `training_results.session_id`)

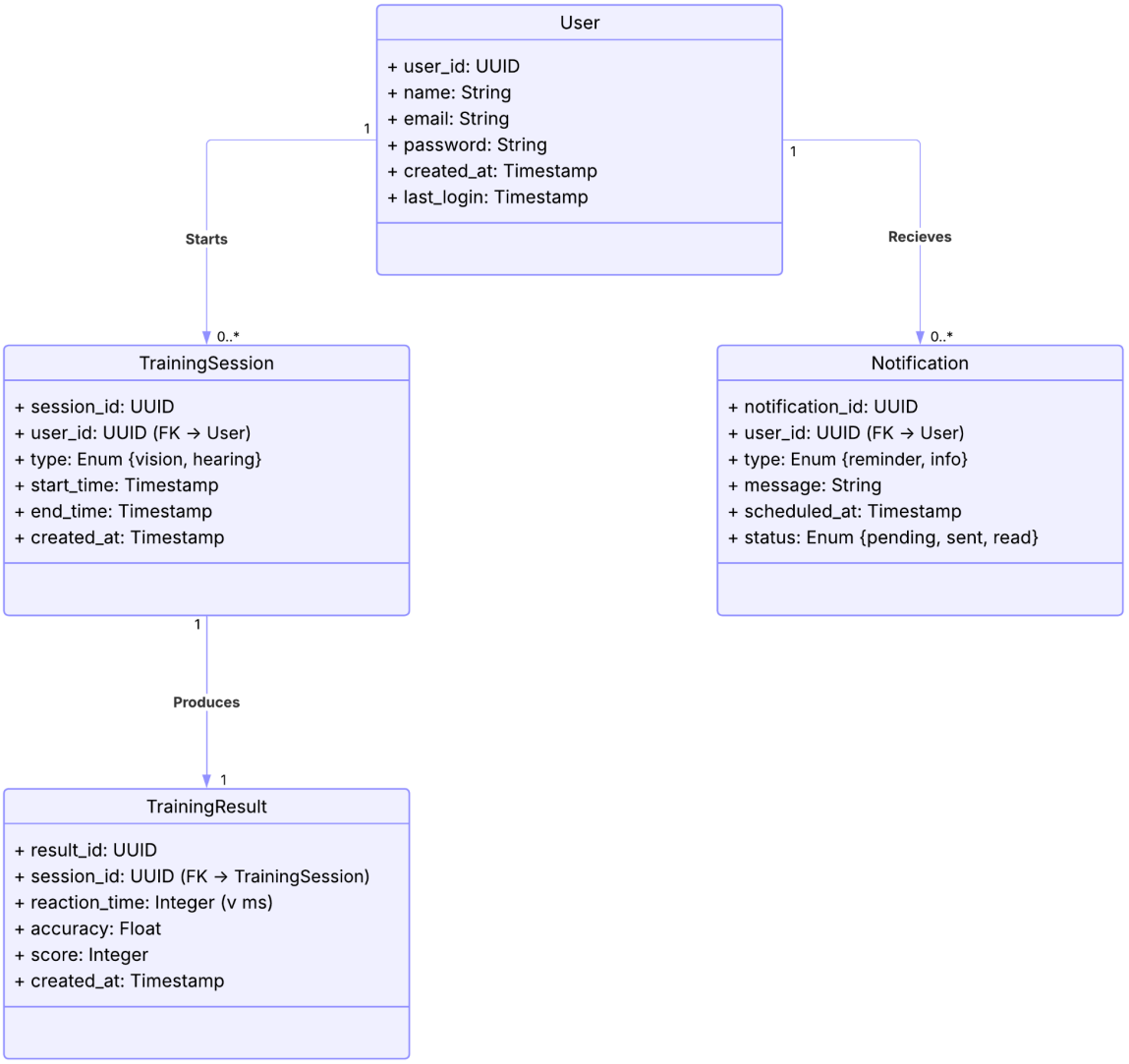
UML diagramy systému

Pro lepší přehlednost a pochopení návrhu systému Sense break byly vytvořeny UML diagramy. Diagramy pokrývají základní strukturu systému, vztahy mezi hlavními entitami, architekturu komponent a hlavní scénáře použití.

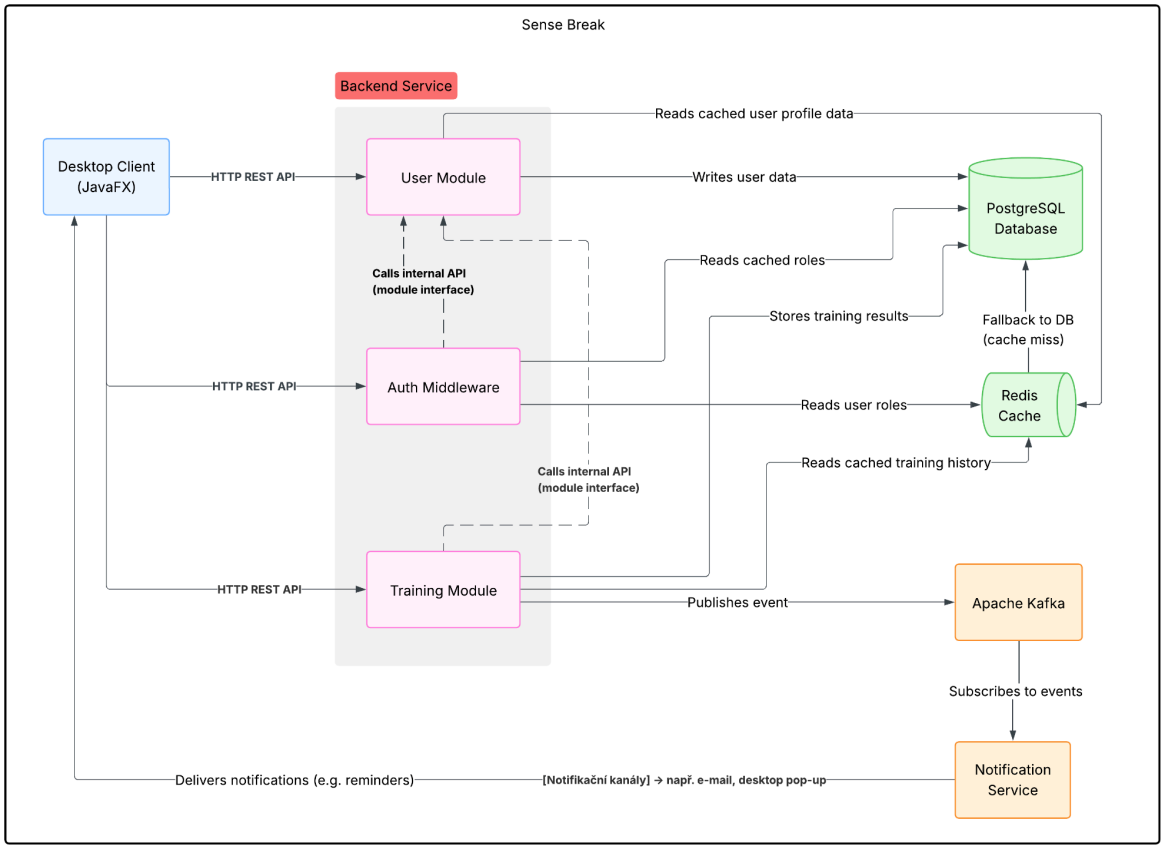
Přehled diagramů

Diagram	**Popis**
Class Diagram	Struktura hlavních entit systému (User, TrainingSession, TrainingResult, Notification) a jejich vztahů.
Component Diagram	Rozložení systému na hlavní komponenty (Desktop klient, mikroslužby, databáze, messaging) a jejich komunikaci.
Sequence Diagram – User Service	Registrace a přihlášení uživatele – komunikace mezi Desktop klientem, User Service a databází.
Sequence Diagram – Training Service	Spuštění tréninku – vytvoření a uložení metadat tréninku.
Sequence Diagram – Result Service	Uložení výsledku tréninku – validace a uložení výsledku do databáze.
Sequence Diagram – Notification Service	Odeslání notifikace – příjem zprávy z RabbitMQ a předání upozornění uživateli.

1. Class Diagram

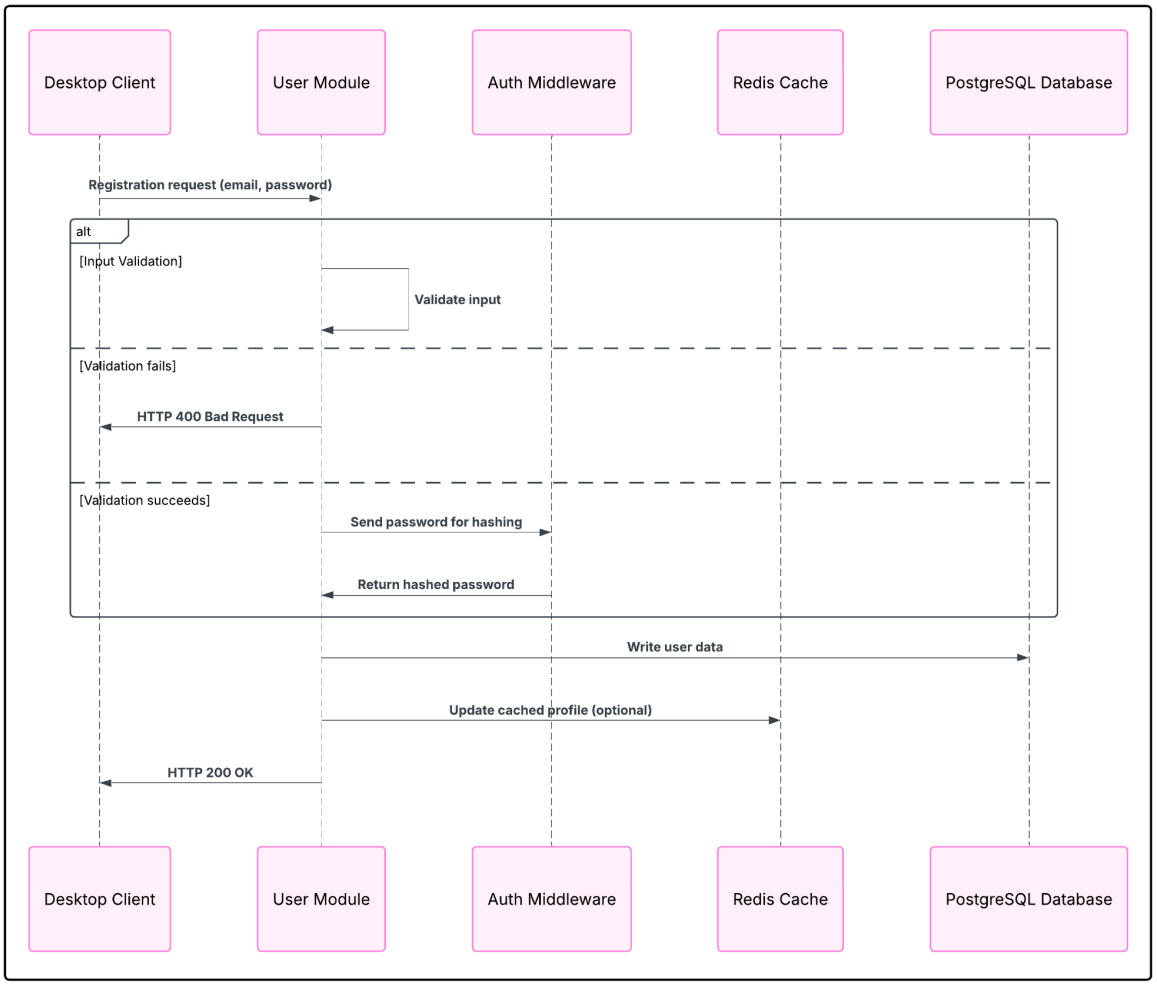


2. Component Diagram

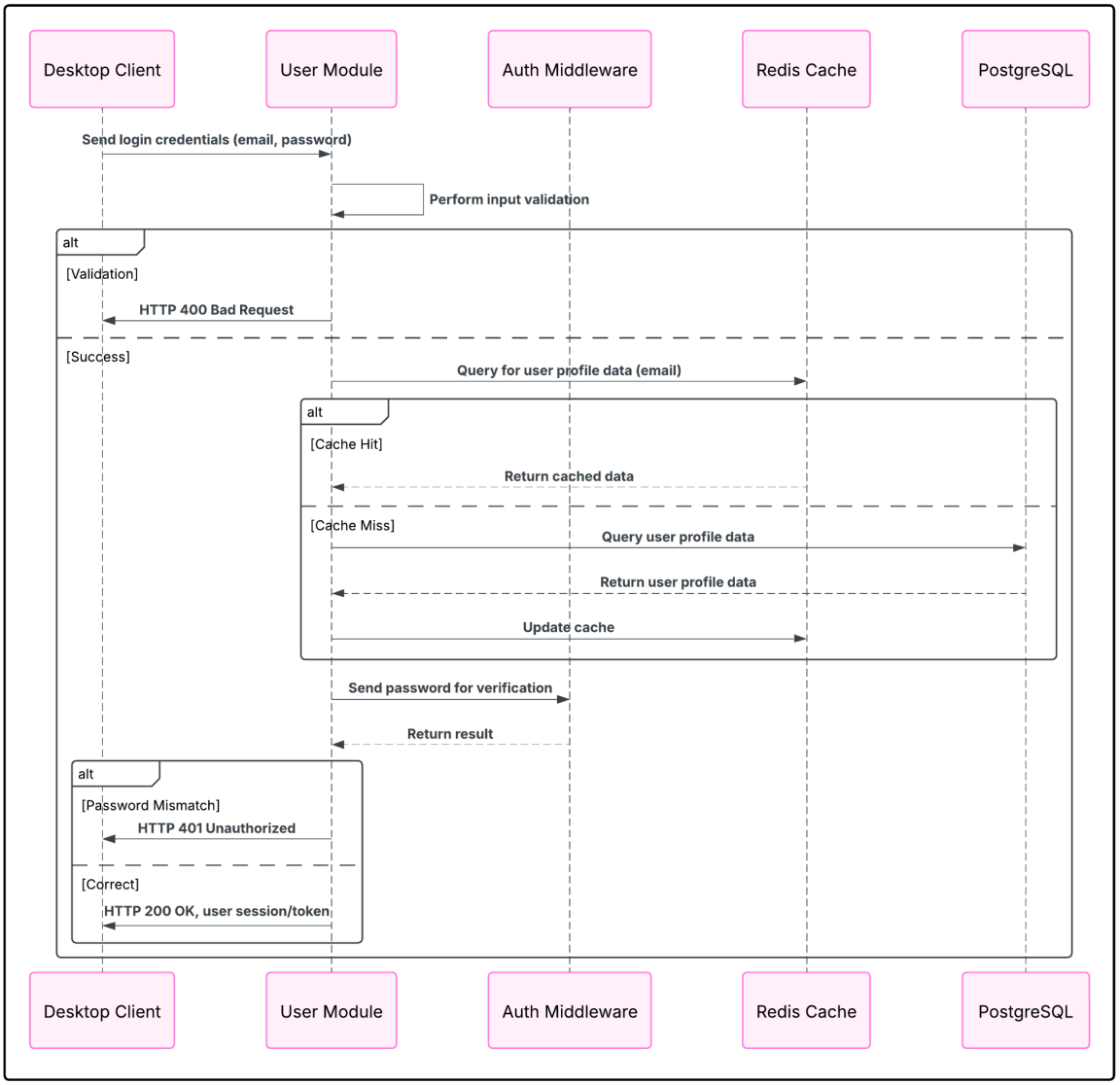


3. Sequence Diagrams:

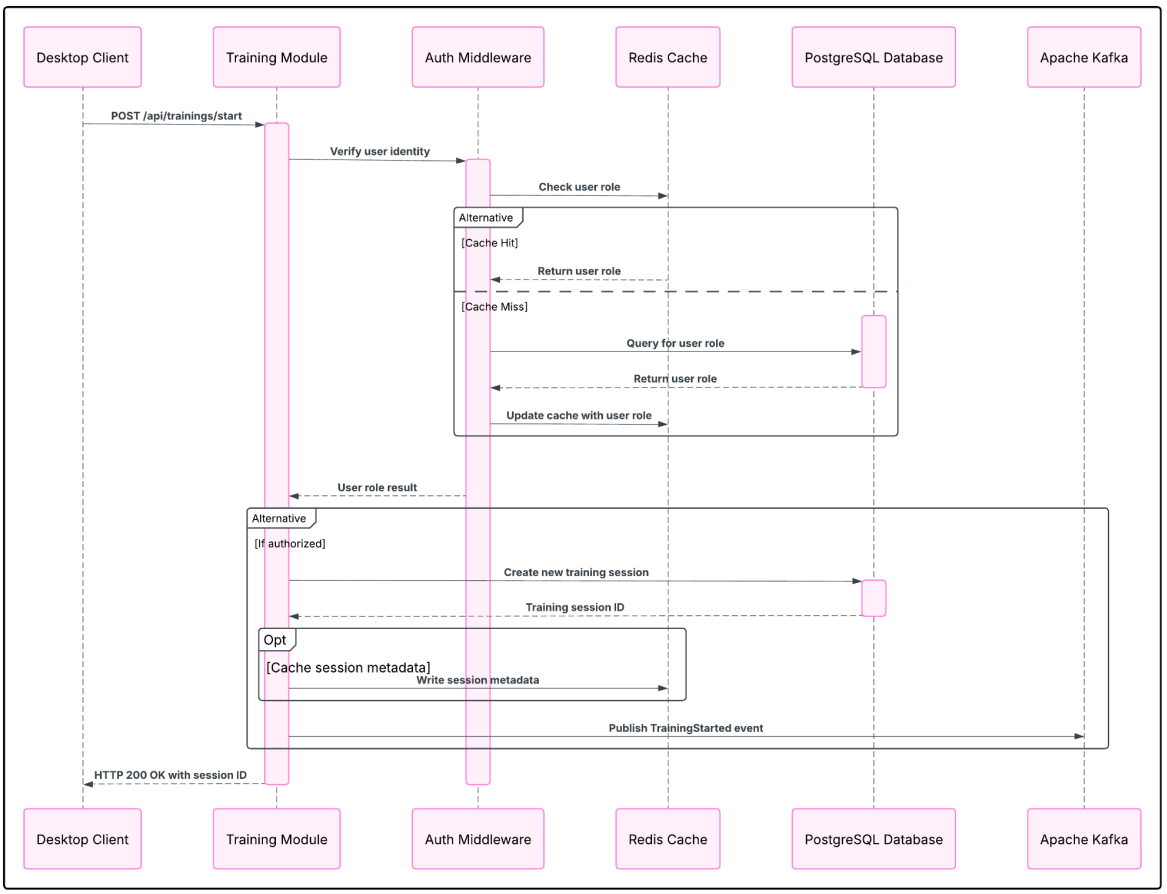
User Module Registration



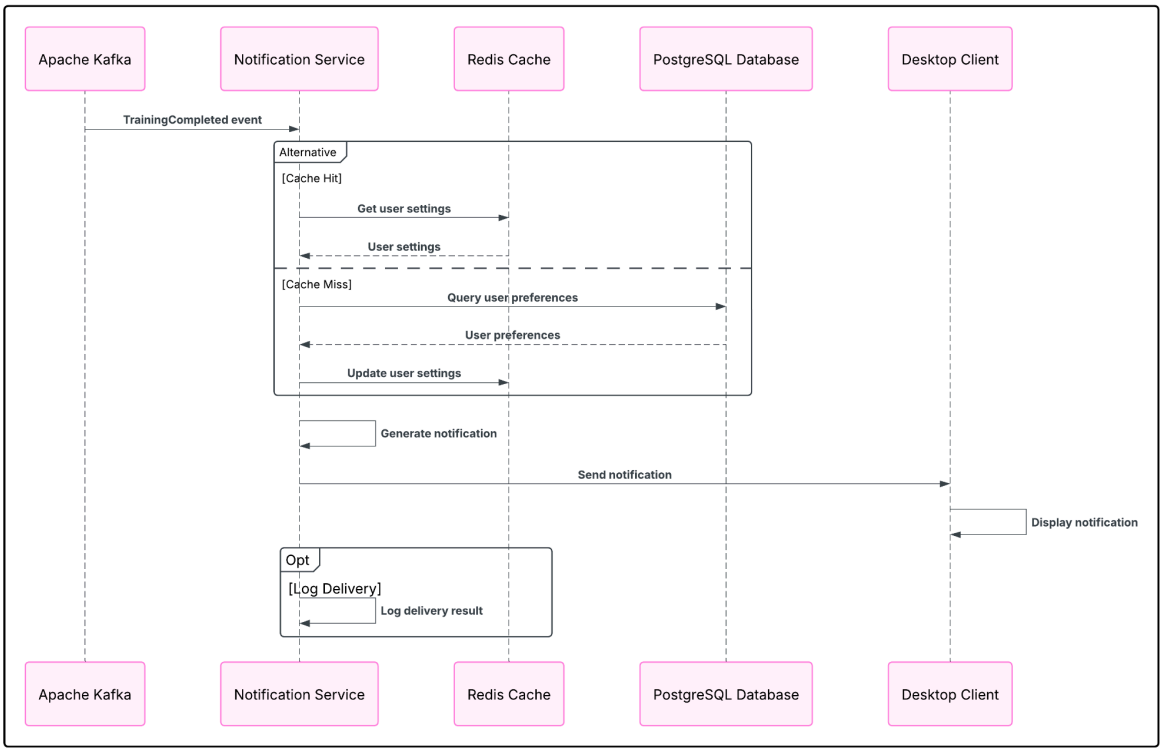
User Mobile Login



Training Module Start Session



Notification Service



Funkční požadavky

Definují, **co systém musí dělat** – tedy hlavní funkcionalitu.

ID	Funkční požadavek	Popis
F1	Registrace uživatele	Uživatel se může zaregistrovat zadáním jména, e-mailu a hesla. Probíhá validace vstupu a uložení dat do databáze a cache.
F2	Přihlášení uživatele	Registrovaný uživatel se může přihlásit do systému. Používá se ověření hesla a čtení uživatelských dat z cache (Redis) nebo databáze.
F3	Spuštění tréninku	Uživatel může spustit zrakový nebo sluchový trénink. Systém ověří oprávnění uživatele (přes Auth Middleware), uloží metadata a pošle událost do Apache Kafka.
F4	Uložení výsledku tréninku	Systém uloží výsledky tréninku (např. reakční čas, skóre, přesnost) do databáze a aktualizuje cache.
F5	Zobrazení historie tréninků	Uživatel může zobrazit historii svých tréninků. Data se načítají z cache a v případě potřeby z databáze.
F6	Plánování notifikací	Uživatel si může naplánovat připomenutí tréninku podle preferencí (čas, typ upozornění).
F7	Odeslání notifikace	Notification Service zpracuje událost z Apache Kafka, načte preferenci uživatele (cache/db), vygeneruje notifikaci a odešle ji uživateli (např. desktop popup).

Nefunkční požadavky

Definují **jak systém funguje** – výkonnost, bezpečnost, spolehlivost atd.

ID	Nefunkční požadavek	Popis
NF1	Hashování hesel	Hesla musí být bezpečně uložena pomocí hashovacího algoritmu (např. BCrypt).
NF2	Autentizace	Přístup k API je chráněn pomocí ověření uživatele (bez RBAC).
NF3	Dostupnost systému	Systém by měl být dostupný 24/7 a zvládat běžné výpadky jedné komponenty.
NF4	Responzivní GUI	Desktopová aplikace musí být přehledná a použitelná na různých velikostech obrazovky.
NF5	Škálovatelnost backendu	Backend musí být připraven na škálování pomocí mikroslužeb a Kafka pro asynchronní komunikaci.
NF6	Zabezpečení	Ochrana proti běžným útokům (např. XSS, CSRF, SQLi).
NF7	Rychlá odezva systému	Odezva API musí být do 500 ms při běžném zatížení. Použití Redis Cache snižuje odezvu při opakovaných požadavcích.
NF8	Zálohování	Data v databázi musí být pravidelně zálohována a obnovitelná.
NF9	Zpracování chyb	API musí vracet konzistentní chybové hlášky (např. 400, 401, 404, 500).
NF10	Výkon cache	Systém musí využívat Redis Cache pro rychlý přístup k často používaným datům (uživatelé, tréninky).
NF11	Monitoring a logging	Každý požadavek a událost (např. notifikace) musí být zaznamenán pro účely ladění a auditování.