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 mikroslužby
2. **Spring Boot** – framework pro tvorbu REST API a backendových služeb
3. **Spring Web** – obsluha HTTP požadavků a definice REST endpointů
4. **Spring Data JPA (Hibernate)** – ORM pro přístup k databázi

Desktopová aplikace

1. **JavaFX** – grafické uživatelské rozhraní pro spuštění 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 a výsledků cvičení

Messaging

1. **RabbitMQ** – message broker pro asynchronní zpracování (např. notifikace)
2. **Spring AMQP** – knihovna pro propojení Spring Boot aplikací s RabbitMQ

Verzování a automatizace

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

Architektura projektu

Typ architektury

Mikroslužby – každá klíčová funkce systému (uživatelé, tréninky, notifikace) je implementována jako samostatná služba, která běží nezávisle a komunikuje přes REST API nebo messaging.

Komunikace

Desktopová aplikace (JavaFX) komunikuje s backendovými mikroslužbami prostřednictvím HTTP požadavků (REST API). Mezi některými službami (např. pro notifikace) probíhá komunikace asynchronně pomocí zpráv přes RabbitMQ.

Hlavní vrstvy backendové mikroslužby:

1. **Controller Layer** – zpracovává HTTP požadavky, volá příslušné služby a vrací odpovědi
2. **Service Layer** – obsahuje aplikační logiku a validaci dat
3. **Repository Layer** – přímý přístup k databázi pomocí Spring Data JPA
4. **Simple Auth Layer** – jednoduchá vrstva pro ověření identity uživatele (např. kontrola ID nebo session)
5. **Messaging Layer** – pro příjem a odesílání zpráv pomocí RabbitMQ (Notification Service)

Struktura desktopového klienta

1. Rozdělení do obrazovek (scén), řídicích tříd (controllers) a FXML souborů
2. Použití JavaFX komponent a layoutů pro zobrazení cvičení a výsledků
3. HTTP komunikace s mikroslužbami pomocí Java HTTP klienta

Messaging

1. **RabbitMQ** – přenáší zprávy mezi službami (např. notifikace po delší nečinnosti)
2. **Asynchronní zpracování** – služby mohou zpracovávat události bez nutnosti přímé synchronní vazby

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

Popis služeb systému

1. User Service

Popis: Spravuje registraci, přihlášení a správu základních informací o uživateli.

Funkcionality:

- Registrace nových uživatelů (`POST /api/auth/register`)
- Přihlášení uživatelů (`POST /api/auth/login`)
- Získání a aktualizace profilu přihlášeného uživatele (`GET/PUT /api/users/me`)

Použité technologie: Java 17, Spring Boot, PostgreSQL

2. Auth Middleware

Popis: Jednoduchá vrstva ověření, že uživatel je přihlášený (např. kontrola session nebo identifikátoru).

Funkcionality:

- Ověření, že požadavek obsahuje platný identifikátor uživatele
- Odmítnutí přístupu při chybějící nebo neplatné identifikaci (`401 Unauthorized`)

Použité technologie: Java 17, Spring Boot

3. Training Service

Popis: Zpracovává logiku tréninků (zrakových a sluchových), zaznamenává výsledky a historii relací.

Funkcionality:

- Spuštění nového tréninku (`POST /api/trainings/start`)
- Zaznamenání výsledku tréninku (`POST /api/trainings/:id/result`)
- Načtení historie tréninků (`GET /api/trainings/history`)
- Získání detailu relace (`GET /api/trainings/:id`)

Použité technologie: Java 17, Spring Boot, PostgreSQL, Spring Data JPA

4. Notification Service

Popis: Zodpovídá za odesílání notifikací uživatelům, např. připomenutí plánovaného tréninku.

Funkcionality:

- Plánování a odesílání připomínek (`SEND` zpráva přes RabbitMQ)
- Příjem asynchronních požadavků od jiných služeb
- Správa notifikační fronty

Použité technologie: Java 17, Spring Boot, RabbitMQ, Spring AMQP

5. Desktop GUI (JavaFX klient)

Popis: Uživatelské rozhraní aplikace pro spouštění tréninků a zobrazení výsledků.

Funkcionality:

- Přihlášení/registrace uživatele
- Spuštění zrakového nebo sluchového cvičení
- Zobrazení historie výsledků
- Odesílání požadavků na REST API

Použité technologie: JavaFX, FXML, Java HTTP Client (JDK 11+)

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. Systém ověří validitu údajů.
3. Systém vytvoří nového uživatele v databázi s výchozí rolí.
4. Uživatel je automaticky přihlášen nebo přesměrován na přihlášení.

Výsledek: Uživatel je zaregistrován a připraven používat aplikaci.

Služby: User Service, Auth Middleware

2. Přihlášení uživatele

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

Kroky:

1. Uživatel zadá e-mail a heslo.
2. Systém ověří, že údaje odpovídají uloženému záznamu.
3. Uživatel je přihlášen a přesměrován na hlavní obrazovku.

Výsledek: Uživatel získá přístup ke svému účtu.

Služby: User Service, Auth Middleware

3. Spuštění tréninku

Popis: Uživatel spustí cvičení zaměřené na zrak nebo sluch.

Kroky:

1. Uživatel si vybere typ tréninku (zrakový / sluchový).
2. Systém vytvoří záznam nové tréninkové relace.
3. Desktopová aplikace zobrazí trénink a sleduje výsledek.
4. Po ukončení se výsledek odesílá na server.

Výsledek: Tréninková relace je zaznamenána v systému.

Služby: Training Service, Desktop GUI

4. Uložení výsledku tréninku

Popis: Po skončení tréninku je výsledek uložen do databáze.

Kroky:

1. Klient odešle metriky (např. skóre, čas) na endpoint pro výsledky.
2. Systém vytvoří nový záznam výsledku a propojí ho s relací.
3. Výsledek je uložen a připraven k zobrazení.

Výsledek: Systém uchová výsledek daného tréninku pro pozdější analýzu.

Služby: Training Service

5. Připomenutí tréninku

Popis: Systém odešle uživateli upozornění, že je čas na další trénink.

Kroky:

1. Jiná služba (např. plánovač) odešle zprávu do RabbitMQ.
2. Notification Service zpracuje zprávu a připraví notifikaci.
3. Uživatel je informován (např. zobrazením zprávy v GUI).

Výsledek: Uživatel obdrží upozornění na trénink.

Služby: Notification Service

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`)

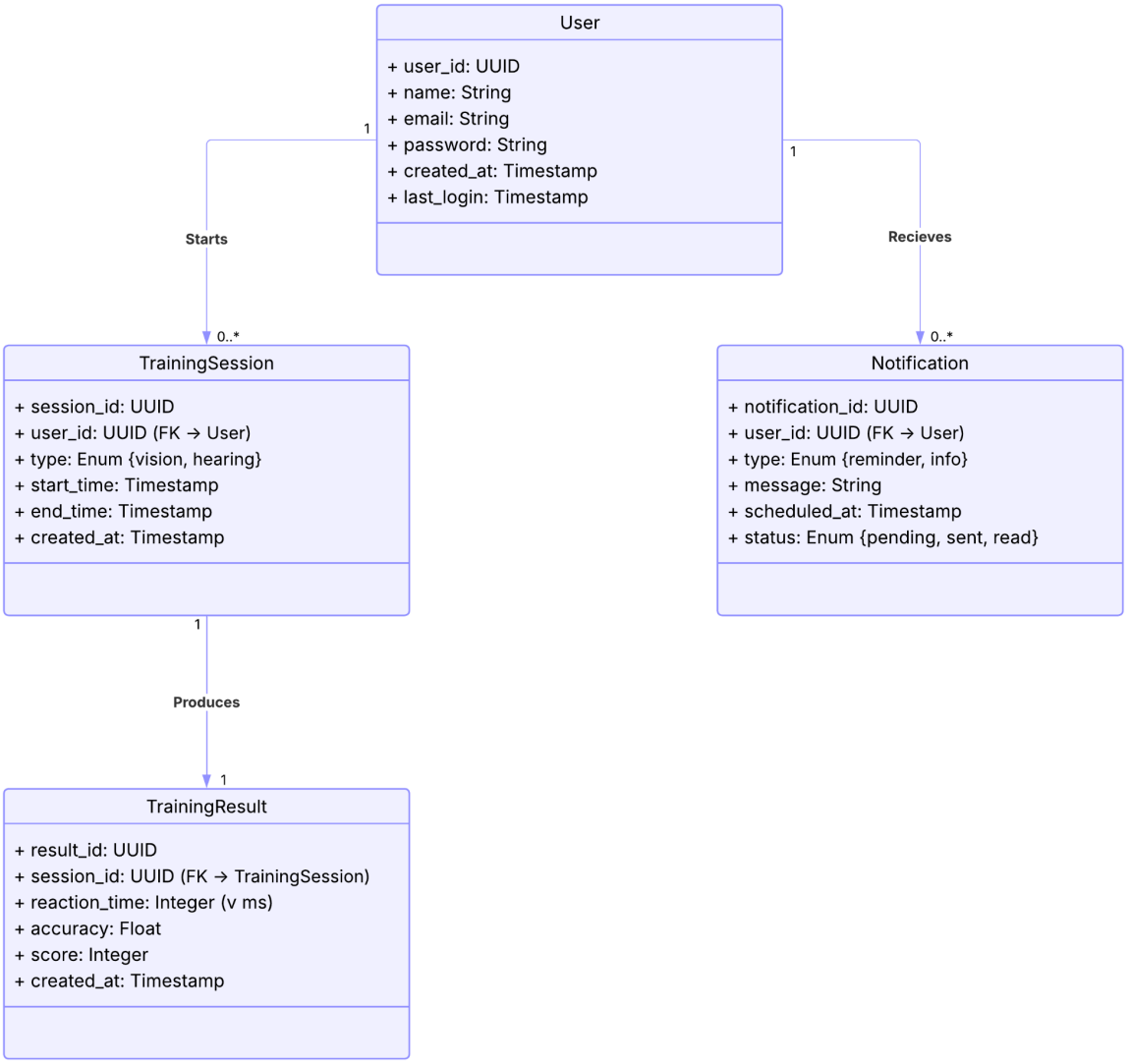
UML diagramy systému

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.

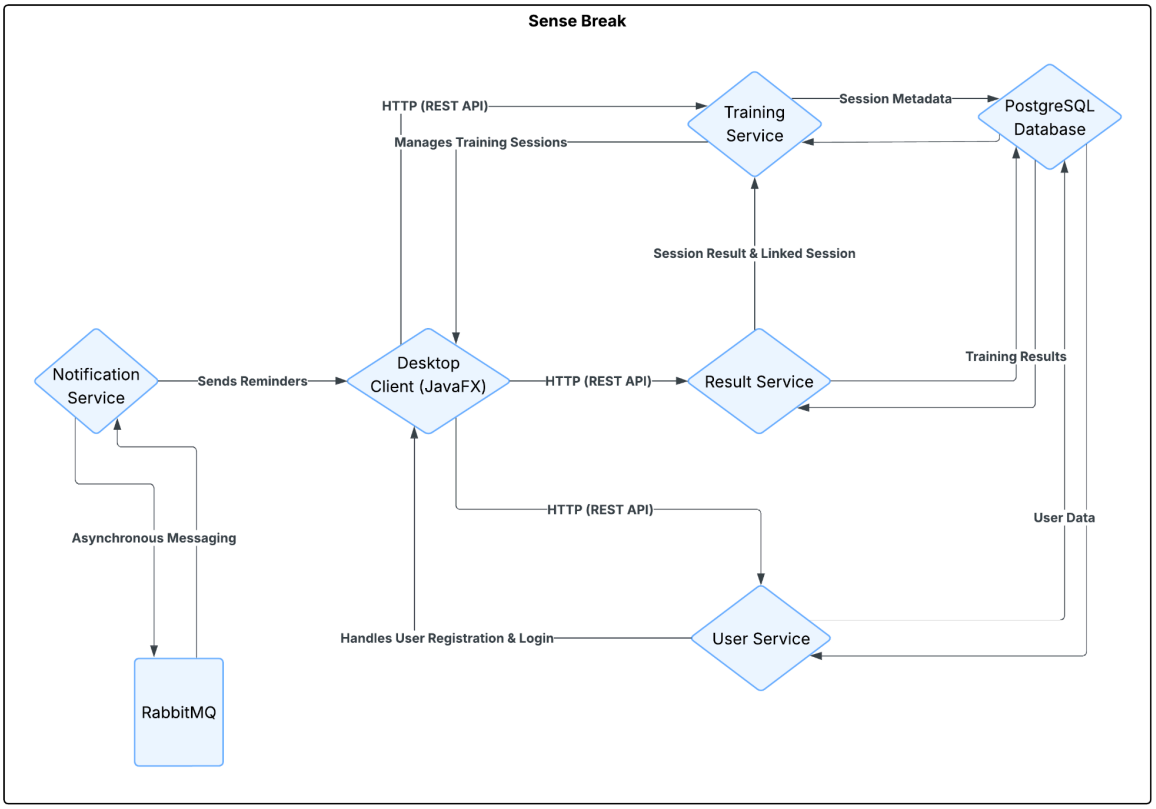
Přehled diagramů

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

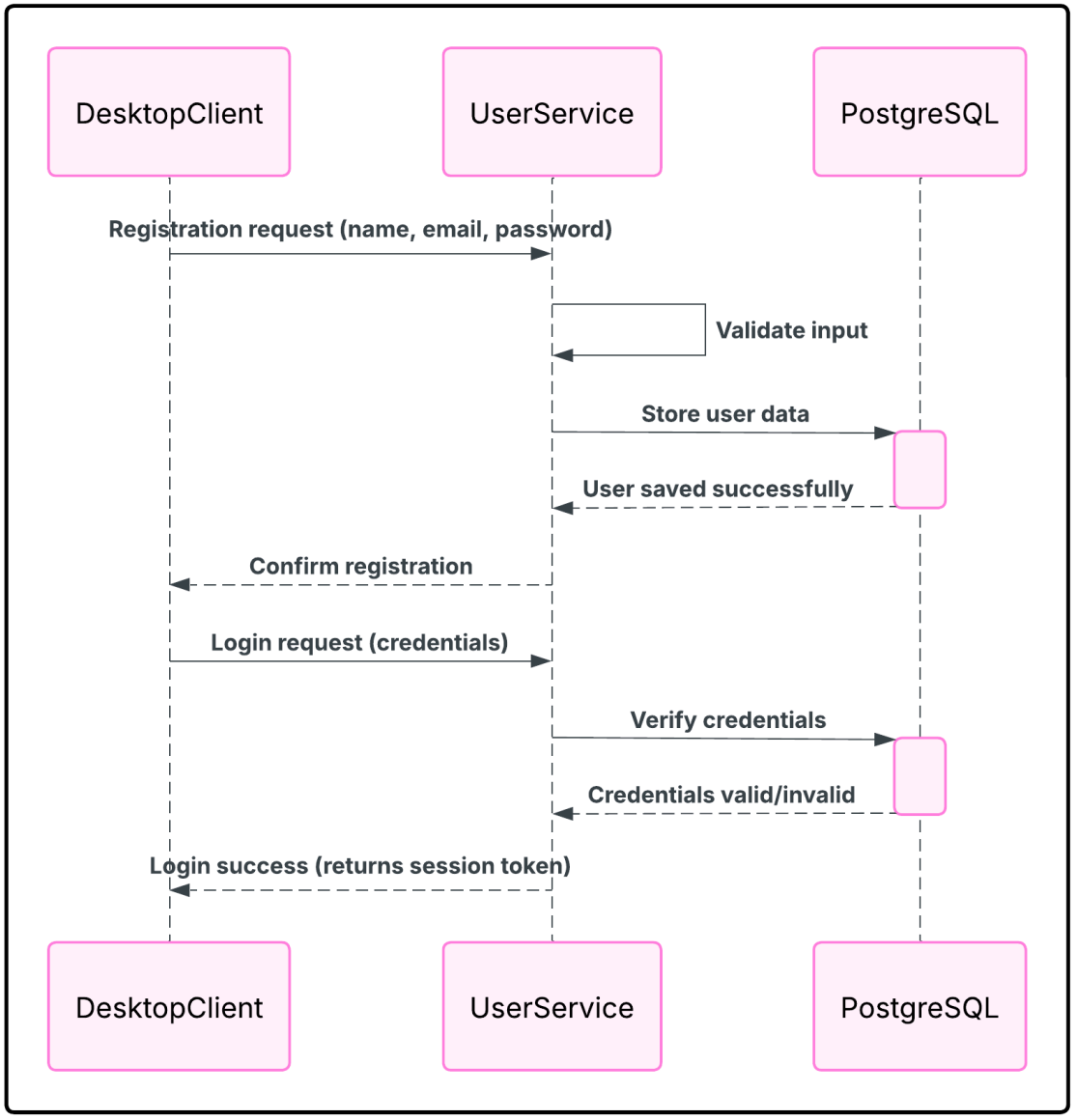


2. Component Diagram

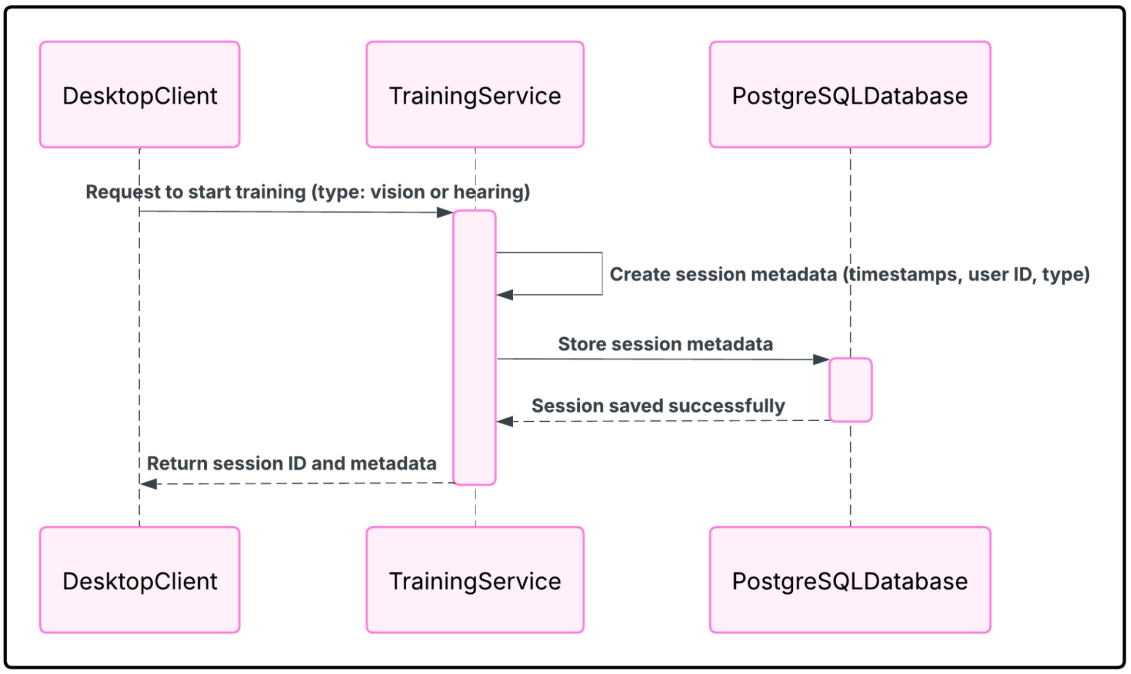


3. Sequence Diagrams:

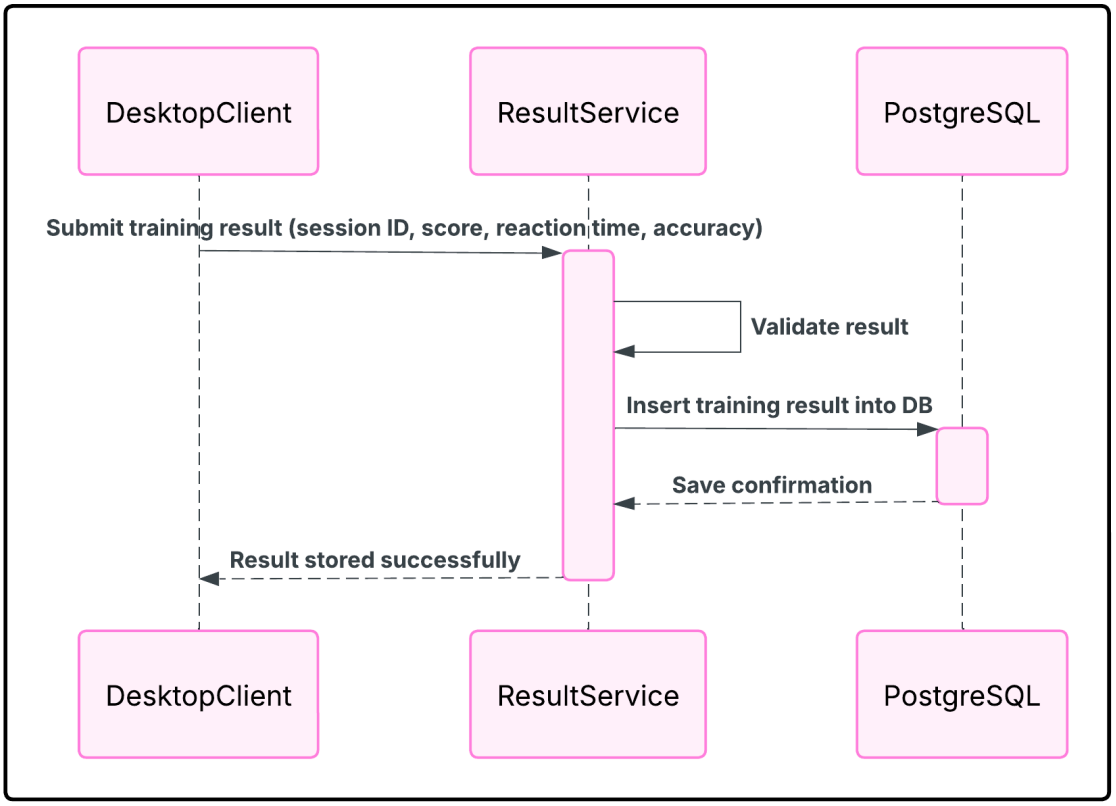
User Service



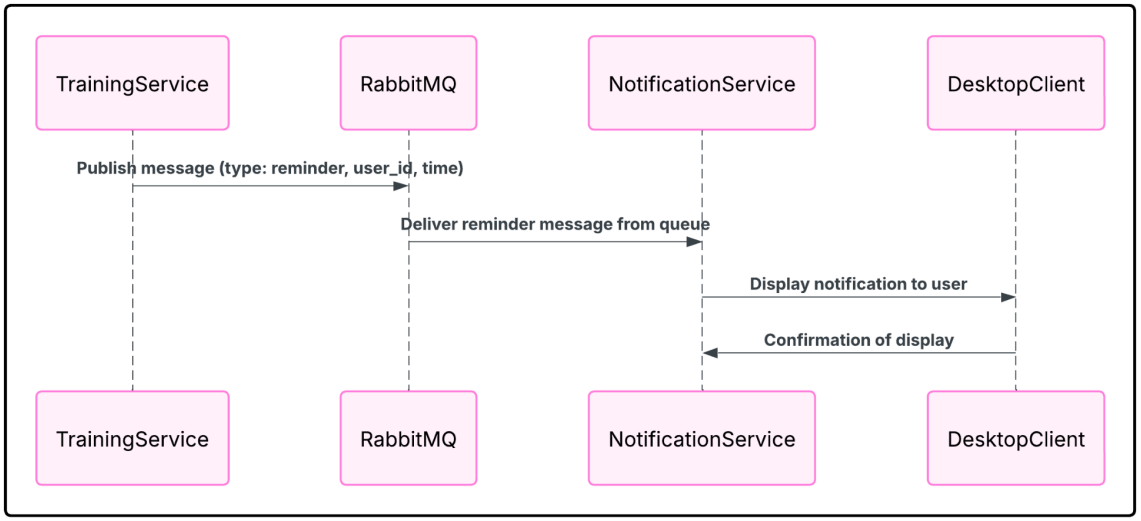
Training Service



Result Service



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.
F2	Přihlášení uživatele	Registrovaný uživatel se může přihlásit do systému.
F3	Spuštění tréninku	Uživatel může spustit zrakový nebo sluchový trénink.
F4	Uložení výsledku tréninku	Systém uloží výsledky tréninku (reakční čas, skóre, přesnost).
F5	Zobrazení historie tréninků	Uživatel může zobrazit historii svých dokončených tréninků.
F6	Plánování notifikací	Uživatel může naplánovat připomenutí na trénink.
F7	Odeslání notifikace	Systém pošle uživateli připomenutí skrze Notification Service.

Nefunkční požadavky

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

ID	Nefunkční požadavek	Popis
NF1	Hashování hesel	Hesla musí být uložena bezpečně pomocí hashovacího algoritmu.
NF2	Autentizace	Přístup k API je chráněn pomocí ověření uživatele (bez RBAC).
NF3	Dostupnost systému	Systém by měl být dostupný 24/7.
NF4	Responzivní GUI	Desktopová aplikace musí být přehledná a funkční na různých rozlišeních.
NF5	Škálovatelnost backendu	Backend by měl být připraven běžet v mikroslužbové architektuře.
NF6	Zabezpečení	Ochrana proti běžným útokům (např. XSS, CSRF).
NF7	Rychlá odezva systému	Odezva API musí být < 500 ms při běžném zatížení.
NF8	Zálohování	Data v databázi musí být pravidelně zálohována.
NF9	Zpracování chyb	API musí vracet konzistentní chybové hlášky (4xx, 5xx).