# **Helpdesk**

Jakub Najman, Jakub Oliberius

Postman testy: https://app.getpostman.com/join-team?invite\_code=51f0b25ba1516884057a8aea76437d43&target\_code=d5aad37160b47be7cfbab4d670187a0e

# Cílový uživatelé

Helpdesk je určeno pro zlepšení komunikace mezi zákazníkem a firmou. Helpdesk ušetří čas tím, že zefektivní organizaci práce a řešení problémů uvnitř firmy, díky němuž se bude zákazník cítit, že je o něj postaráno.

Helpdesk bude obsahovat 2 skupiny uživatelů – zákazníky (CUSTOMER) a zaměstnance firmy, kteří se dále dělí podle rolí ve firmě (COMMON, MANAGER, ...). Pravomoce uživatelů k manipulaci s požadavky jsou následující:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Vytvářet / create | Číst / read | Změnit / update | Smazat / delete |
| CUSTOMER | √ | √ | √ | √ |
| MANAGER | Ｘ | √ | √ | √ |
| EMPLOYEE (common) | Ｘ | √ | √ | X |

# Funkce projektu

Zákazníci budou schopni vytvářet nové požadavky (requesty), které manažer firmy budě přidělovat zaměstnancům k vyřešení.

Zákazníci budou zaměstnanci informováni o vyřešení jejich požadavku a budou muset označit požadavek za vyřešený, před jejím odstraněním (zavřením), které provádí manažer.

Uživatelé budou moci komentovat požadavky.

Zaměstnanci si budou měřit čas strávený prací na určitém požadavku a budou moci informovat uživatele o jeho dokončení/vyřešení.

Helpdesk umí zobrazit požadavky podle různých filtrů (datum, závažnost, typu atd…)

# Omezení

**Správa požadavků pouze v rámci firmy:** Helpdesk je navržený pro interní komunikaci mezi zákazníky a firmou. Neposkytuje podporu pro externí komunikaci nebo veřejné dotazy.

**Omezená role zákazníků:** Zákazníci mají omezenou pravomoc ve změně nebo mazání požadavků, což zabraňuje zneužití.

**Povinnost zákazníků označit požadavek jako vyřešený:** Zákazníci musí schválit vyřešení požadavku, protože bez schválení nelze požadavek označit za vyřešený.

**Omezená práce zaměstnance závislá na typu oddělení**: Jen zaměstnanci z určitého oddělení mohou pracovat pouze na požadavcích zahrnující toto oddělení.

# UML Diagram – model

A diagram of a computer

Description automatically generated

# Struktura aplikace

Aplikace je rozdělena do 4 vrstev (Model, DAO, Service, Controller). DAO vrstva je vrstva, která přistupuje do databáze a hledá určité třídy modelu v databázi. Service vrstva obsahuje bussiness logiku a Controllery vystavují RESTové rozhraní dostupné přes endpointy. Použitá databáze je PostgreSQL a k nasazení a spuštění aplikace byl použit Docker.

Aplikace spustíte tak, že si pomocí dockerfilu, který je přístupný v projektu vytvoříte image, který lze následně spustit v kontejneru dockeru.

# Získané zkušenosti

Při práci na tomto projektu jsme si vyzkoušeli spoustu technologií, jako je například Spring, Springboot, Spring data JPA nebo Spring security, která nám dělala největší problém. Security nám dělala problém, protože v našem modelu máme třídu User jako abstraktní třídu, ze které dědí Customer a Employee. To se ze začátku zdálo jako dobrý nápad a kvůli naší programátorské nezkušenosti jsem do našeho projektu tuto chybičku vepsali. Když přišel čas implementace security a přihlašování, tak jsme zjistil, že není snadné udělat jednotné přihlášení pro dvě různé entity a museli jsme to obejít tím, že při přihlášení procházíme napřed tabulku Customer a snažíme se uživatele přihlásit, a to když nevyjde tak projdeme ještě tabulku Employee. Když alespoň v jednom případě najdeme schodu, tak ale naše UserDetails vytvořené po příhlášení musí mít oba atributy jak Customer tak Employee a pouze jeden z nich bude null a role se správně nastaví. Tento způsob sice řeší náš problém, ale to jen za předpokladu, že emailová adresa použitá k přihlášení Employee nemůže být nikdy použita k přihlášení Customera. Dále jsme si hodně procvičili psaní testů jak s pomocí Springu tak s pomocí Mockito frameworku.  
Celkově byly zkušenosti získané za jeden semestr k nezaplacení a nemůžu uvěřit, že tento předmět už není povinný pro všechny obory a například předmět jako C++ je.