Система за съседски обмен

Екипа

27.10.2024

|  |  |  |  |
| --- | --- | --- | --- |
| Версия № | Дата | Автор | Описание на изменението |
| Версия 1 | 2.11.2024 | Целия екип | Първо издание |

# Въведение

## Този документ описва архитектурата на проекта Neighbors Exchange – уеб приложение за улесняване на обмена на предмети между локални потребители. Той предоставя структуриран подход към дизайна, архитектурата и функционалността, позволявайки на потребителите да добавят обяви и да се свързват с други потребители за обмен.

## Участниците:

### Разработчици: Отговарят за разработката и поддръжката на приложението.

### Крайни потребители: Членове на местната общност, интересуващи се от обмен на предмети.Марин Динчев -

### Администратори: Следят и управляват дейностите на потребителите в платформата.

# Предназначение

## Обхват

## Тозу документ обхваща планирането на софтуерната архитектура на проекта и решенията, който сме взели относно нефункционалните изисквания

## Актьори

### **Обикновени потребители:** Могат да добавят обяви, да преглеждат налични предмети и да организират обмен с други потребители.

### **Администратори:** Наблюдават и управляват обявите, обработват докладвани предмети и подпомагат управлението на потребителите**.**

## Този документ е насочен към преподавателите по предмета „Софтуерни Технологии“.

## Използвани термини и символи

1. API (Интерфейс за програмно свързване): Определя методите за комуникация между фронтенда и бекенда.
2. DB (База данни): В случая PostgreSQL, използвана за съхранение на данните на приложението.
3. ORM (Обектно-релационно моделиране): Метод за свързване на данни между обекти и релационни бази данни, в случая с Prisma.
4. Clerk: Външна услуга, използвана за удостоверяване на потребителите и тяхното управление.

## Източници

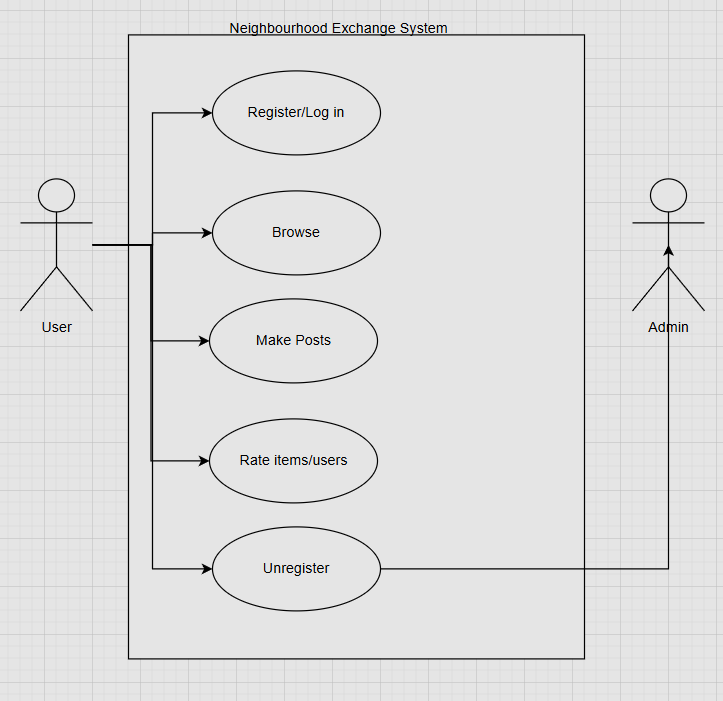
### Официална документация на Remix

### Документация на Prisma

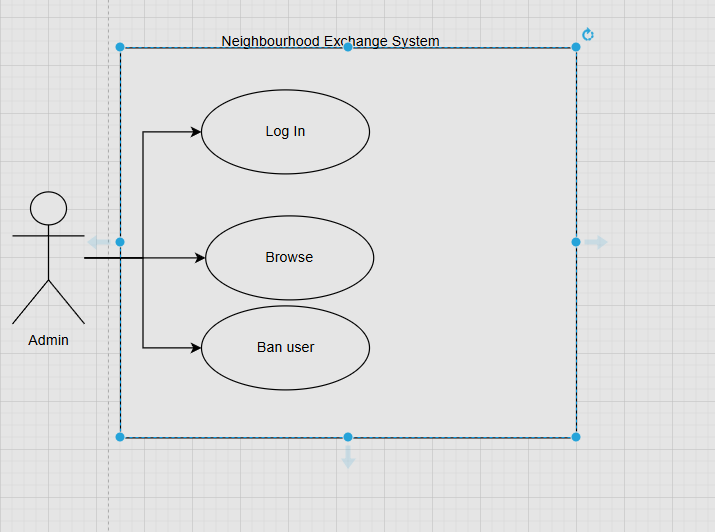
### Документация на Clerk за удостоверяване

# Архитектурен обзор

## Use-case изглед

Тази част демонстрира функционалностите, достъпни за потребителите, чрез диаграми на UML Use-case и описания. Основни сценарии на крайния потребител включват регистриране и дерегистриране на потребител, добавяне на обяви, преглеждане на обяви и докладване на неподходящи такива .

## Функционалностите на админа включват влизане в системата, чрез Clerk, разглеждане на съществуващи обяви и забраняването на достъп до системата на потребители направили нарушения.



## Логически изглед

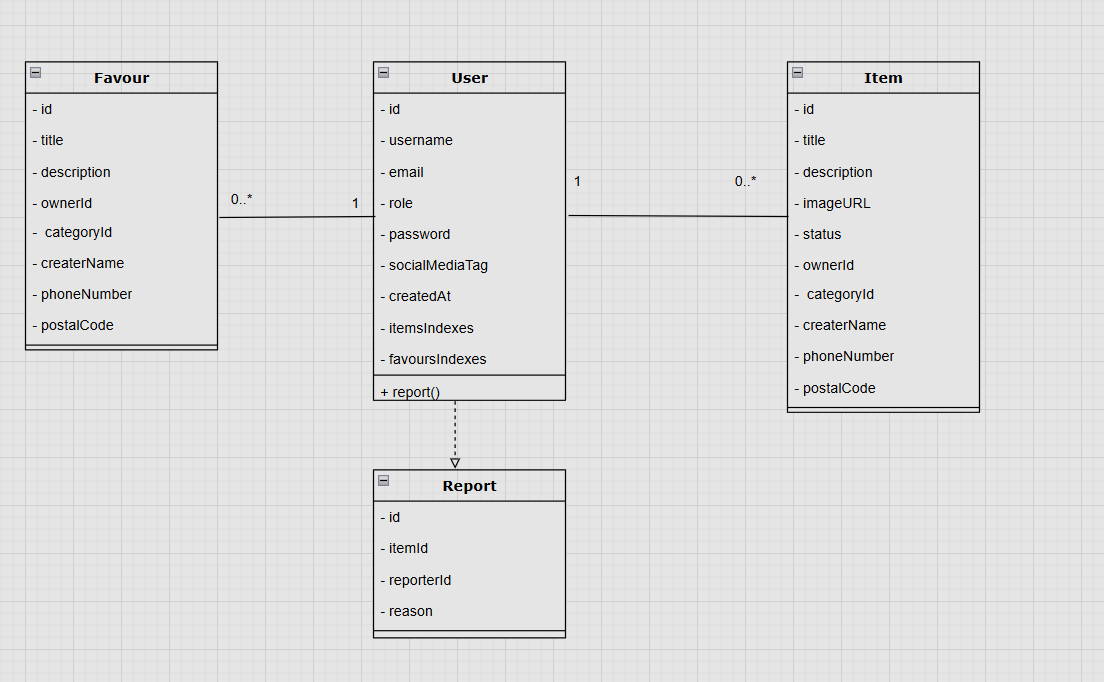
### Логическата структура на приложението включва основни модули и класове. Основните компоненти са:

#### Модул за потребители: Управлява данните за потребителите и техните роли.

#### Модул за предмети: Управлява предметите, добавени за обмен.

#### Модул за категории: Организира предметите по категории.

#### Модул за докладване: Обработва докладваните предмети.



## Процесен изглед

### Добавяне на обява

#### Описание: Потребител създава нова обява за предмет, който желае да обмени или даде безвъзмездно, или за услуга.

#### Процес:

##### Потребителят попълва информация за предмета (заглавие, описание, категория, снимка и други данни).

##### Ако обявата е за услуга, потребителят попълва заглавие, описание и друга информация

##### След потвърждение, системата проверява за валидност на данните и изпраща заявка към базата данни, за да създаде нов запис в таблицата Предмети.

##### Базата данни връща идентификатор на новосъздадената обява, който се показва на потребителя.

### Преглед на обяви

#### Описание: Потребителите могат да разглеждат всички активни обяви в платформата.

#### Процес:

##### Потребителят избира район чрез пощенски код

##### Системата изпраща заявка към базата данни, извличайки обявите, които съответстват на критериите за търсене.

##### Данните се връщат и се показват в интерфейса на потребителя.

### Управление на обмена

#### Описание: Потребителят се свързва с друг потребител за организиране на обмен.

#### Процес:

##### Потребителят избира обява и получава информация за контакт със създателя на обявата.

##### Системата изпраща известие до собственика на обявата чрез външни методи на комуникация (телефон или социални мрежи).

##### Двамата потребители се договарят извън платформата и впоследствие могат да потвърдят обмена чрез системата, която актуализира „История на обмените“.

### Докладване на обява

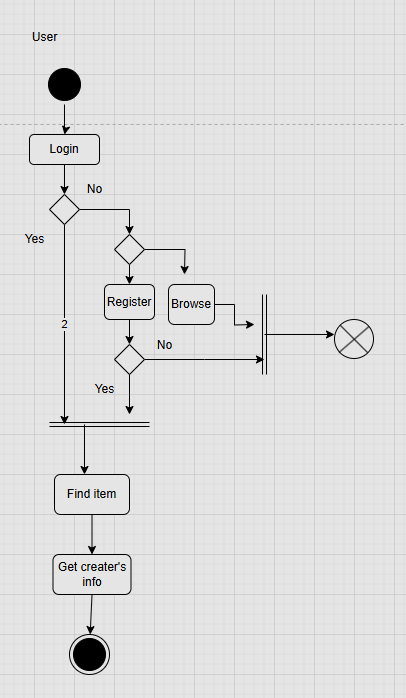
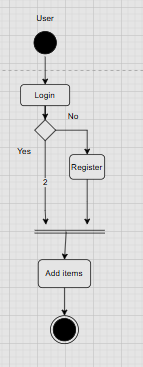
#### Описание: Потребител докладва неподходяща или неправомерна обява.

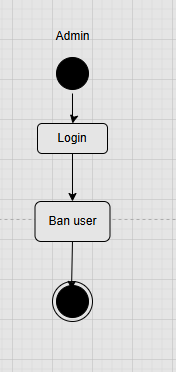
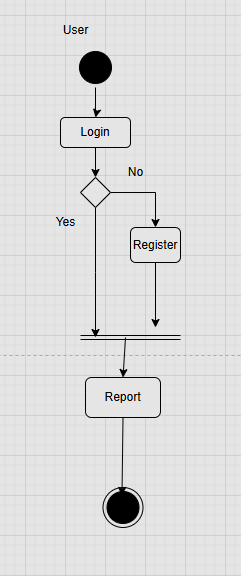
#### Процес:

##### Потребителят избира опцията за докладване и посочва причина.

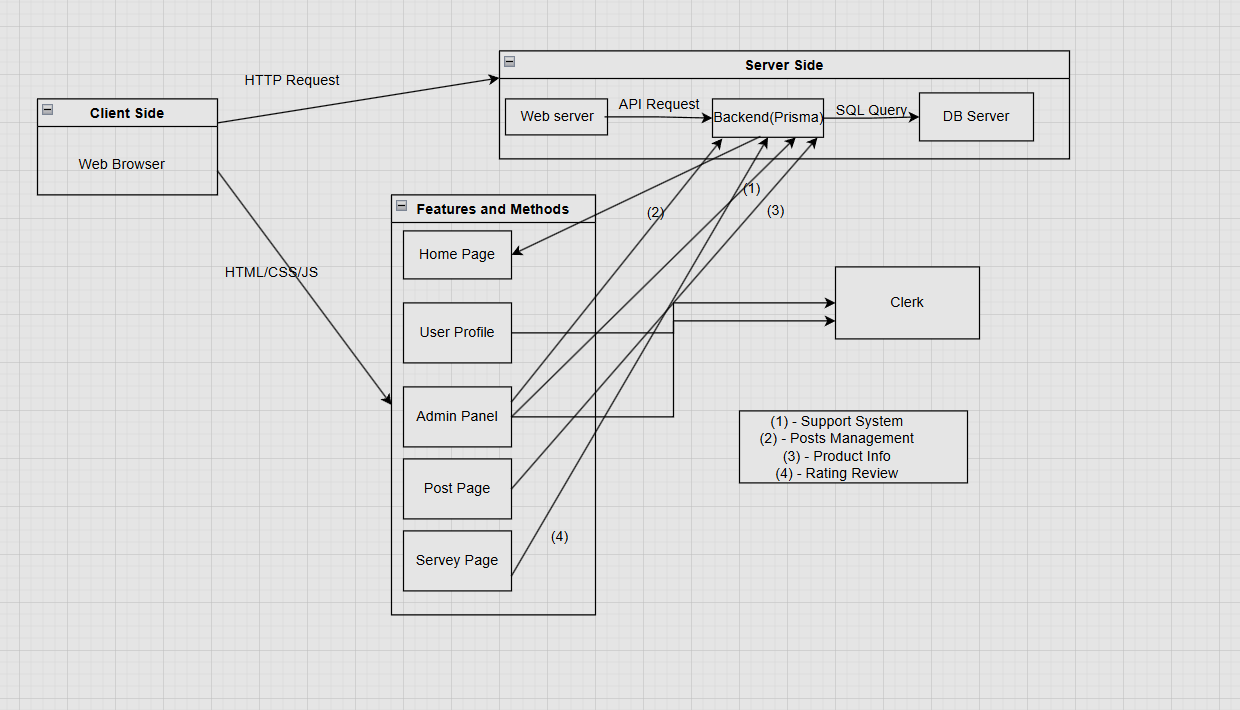
##### Системата изпраща данните към администратора за преглед, създавайки запис в таблицата Доклади.

##### Администраторът получава известие и може да разгледа доклада, и предприеме действия (изтриване на обявата или предупреждение за потребителя).

## Изглед на внедряването

Приложението е внедрено в облачна среда, свързвайки фронтенда (Remix framework) с бекенд API и базата данни PostgreSQL. Clerk предоставя външно удостоверяване на потребителите, улеснявайки управлението на потребителските акаунти. 

### Фронтенд: Изграден с Remix framework, отговаря за потребителския интерфейс и логиката за взаимодействие.

### Бекенд: Предоставя API и бизнес логиката, използвайки Prisma за връзка с базата данни.

### Слой за базата данни: PostgreSQL, достъпван чрез Prisma ORM.

### Слой за удостоверяване: Управляван от Clerk за сигурно влизане, регистрация и управление на роли на потребителите.

# Нефункционални изисквания

## Достъпност: Гарантира минимално прекъсване; потребителите винаги имат достъп до услугата.(4/5)

## Мащабируемост: Проектиран да поддържа нарастващ брой потребители и обяви.(4/5)

## Производителност: Оптимизирани заявки към базата данни, бърз и отзивчив интерфейс.(3/5)

## Сигурност: Използва Clerk за сигурно управление на потребителите, ролево базиран достъп и поверителност на данните.(5/5)

## Интероперативност: Съвместим с модерни браузъри и мобилни устройства.(2/5)

## Използваемост: Удобен за потребителя интерфейс, ясна навигация и бърз достъп до функции.(4/5)