**​ДИПЛОМЕН ПРОЕКТ**

**​ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ**

​

​по професия код 481030 „Приложен програмист“

​специалност код 4810301 Приложно програмиране“

​

**​Тема: „Домашен асистент“**

​

​

​Автор:

​Калоян Динков Динев, клас XII В

​

​Ръководител:

​Антони Димитров

Бургас

СЪДЪРЖАНИЕ

[1 Увод 4](#_Toc132833829)

[2 Цели и обхват на софтуерното приложение 4](#_Toc132833830)

[2.1 Цели 4](#_Toc132833831)

[2.2 Обхват 6](#_Toc132833832)

[3 Анализ на решението 7](#_Toc132833833)

[3.1 Потребителски изисквания и работен процес 7](#_Toc132833834)

[3.1.1 Потребителски изисквания 7](#_Toc132833835)

[3.1.2 Работен процес 9](#_Toc132833836)

[3.2 Примерен потребителски интерфейс 20](#_Toc132833837)

[3.2.1 Локален сървър 20](#_Toc132833838)

[3.3 Диаграми на анализа 22](#_Toc132833839)

[3.4 Диаграма на връзката на обекта 22](#_Toc132833840)

[3.4.1 Локален Сървър 22](#_Toc132833841)

[3.5 Модел на съдържанието / данните 26](#_Toc132833842)

[4 Дизайн 26](#_Toc132833843)

[4.1 Реализация на архитектурата на приложението 26](#_Toc132833844)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 27](#_Toc132833845)

[4.2.1 Диаграми на класовете на локалния сървър 27](#_Toc132833846)

[4.2.2 Диаграми на класовете на уеб приложението 34](#_Toc132833847)

[4.2.3 Диаграми на класовете на BridgeAPI 37](#_Toc132833848)

[4.2.4 Описание на слоевете и тяхната функция 41](#_Toc132833849)

[4.2.5 Комуникация между локания сървър и интернет на нещата устройствата 42](#_Toc132833850)

[4.2.6 Комуникация между BridgeAPI и уеб приложението 43](#_Toc132833851)

[4.2.7 Комуникация между BridgeAPI и локалния сървър 44](#_Toc132833852)

[4.3 Организация и код на заявките към база от данни 46](#_Toc132833853)

[4.3.1 Персонализираната обектно релационен картограф (ORM) система 47](#_Toc132833854)

[4.3.2 Обектно релационен картограф (ORM) – Entity Framework Core 50](#_Toc132833855)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 51](#_Toc132833856)

[5 Ефективност и бързодействие на решението 51](#_Toc132833857)

[6 Тестване 52](#_Toc132833858)

[7 Заключение и възможно бъдещо развитие 54](#_Toc132833859)

[7.1 Заключение 54](#_Toc132833860)

[7.2 Бъдещо развитие 56](#_Toc132833861)

[8 Използвани литературни източници и Уеб сайтове 57](#_Toc132833862)

[9 Приложения 57](#_Toc132833863)

[10 Критерии и показатели за оценяване 57](#_Toc132833864)

# Увод

Тъй като технологията продължава да напредва, виждаме нарастваща тенденция в използването на свързани устройства и технологията интернет на нещата за подобряване на ежедневието ни. Една област, в която тази технология се оказа особено полезна, е домашната сигурност и наблюдение. Използвайки модулна интернет на нещата система за домашна сигурност, собствениците на жилища могат да персонализират настройките си за сигурност, за да отговарят на техните уникални нужди и предпочитания, като същевременно се възползват от свързана мрежа от сензори и устройства, които могат да откриват и предупреждават за потенциални прониквания или опасности за безопасността.

Ползите от интернет на нещата технологията в дома обаче се простират отвъд сигурността. С добавянето на сензори и устройства за наблюдение на здравето, собствениците на жилища могат също така да следят личните си здравни показатели и да получават предупреждения, когато трябва да предприемат действия, за да подобрят благосъстоянието си. Тази интеграция на технологии за сигурност и наблюдение на здравето не само повишава безопасността на дома, но и цялостното качество на живот на неговите обитатели.

В този проект ще проучим дизайна и внедряването на модулна интернет на нещата система за домашна сигурност, която включва както функции за сигурност, така и функции за наблюдение на здравето. Ще разгледаме различните компоненти, които изграждат системата, както и необходимия софтуер и свързаност, за да гарантираме, че всички устройства комуникират ефективно. До края на този проект ще имате задълбочено разбиране за това как да изградите и персонализирате своя собствена модулна интернет на нещата система за домашна сигурност и как да включите функции за наблюдение на здравето, за да създадете един наистина модерен и свързан дом.

# Цели и обхват на софтуерното приложение

## Цели

Целите на модулно интернет на нещата приложение за домашна сигурност и мониторинг на здравето могат да включват:

1. Подобряване на сигурността на дома: Основната цел на приложението за сигурност на дома е да открива потенциални прониквания и да изпраща предупреждения до собственика на дома, което му позволява да предприеме действия за защита на своята собственост и близки. Модулна интернет на нещата система за домашна сигурност позволява по-голяма персонализация и гъвкавост, като позволява на собствениците на жилища да добавят или премахват сензори и устройства, ако е необходимо, за да създадат система, която отговаря на техните уникални нужди.
2. Персонализиране на функциите за сигурност: Като позволява персонализиране на функциите за сигурност, модулната интернет на нещата система за домашна сигурност може да бъде пригодена да отговаря на предпочитанията и начина на живот на собственика на дома. Тази гъвкавост означава, че собствениците на жилища могат да инсталират сензорите и устройствата, които са най-подходящи за тях, осигурявайки им по-голям контрол върху настройките за сигурност.
3. Подобряване на личното здраве: Интегрирането на функции за наблюдение на здравето в модулна интернет на нещата система за домашна сигурност позволява на собствениците на жилища да наблюдават личните си здравни показатели и да получават предупреждения, когато трябва да предприемат действия за подобряване на своето благосъстояние. Това може да включва наблюдение на жизнените показатели, проследяване на моделите на съня или наблюдение за потенциални рискове за здравето.
4. Създаване на свързан дом: Чрез интегриране на модулна интернет на нещата система за домашна сигурност с други устройства и сензори в дома, собствениците на жилища могат да създадат наистина свързан дом. Това позволява по-голяма автоматизация и контрол върху различни аспекти на дома, като осветление, температура и системи за забавление.
5. Осигуряване на спокойствие: В крайна сметка целта на модулно интернет на нещата приложение за домашна сигурност и мониторинг на здравето е да осигури спокойствие на собственика на жилище, знаейки, че домът му е защитен и личното им здраве се наблюдава и проследява в реално време. Чрез създаването на система, която е адаптивна, гъвкава и лесна за използване, собствениците на жилища могат да поемат контрола върху своите нужди за сигурност и наблюдение на здравето, което им позволява да бъдат спокойни, знаейки, че са защитени и се грижат за тях.

## Обхват

Потребителите на модулно интернет на нещата приложение за домашна сигурност и мониторинг на здравето могат да включват:

1. Собственици на жилища: Основната аудитория за модулно интернет на нещата приложение за домашна сигурност и мониторинг на здравето са собствениците на жилища. Собствениците на жилища, които са загрижени за своята сигурност и лично здраве, могат да се възползват от система, която интегрира функции за сигурност и наблюдение на здравето. Чрез персонализиране на системата, за да отговаря на техните уникални нужди и предпочитания, собствениците на жилища могат да бъдат спокойни, знаейки, че техният дом и лично здраве се наблюдават и защитават.
2. Членове на семейството: Членовете на семействата на собствениците на жилища също могат да се възползват от модулна интернет на нещата система за домашна сигурност и мониторинг на здравето. Например, възрастни родители, живеещи сами, могат да се възползват от функции за наблюдение на здравето, които откриват потенциални рискове за здравето и изпращат предупреждения до членове на семейството или лица, които се грижат за тях. По същия начин децата или тийнейджърите могат да се възползват от функциите за сигурност на системата, осигурявайки на родителите спокойствие, знаейки, че децата им са в безопасност.
3. Полагащите грижи: Полагащите грижи също могат да се възползват от модулна интернет на нещата система за домашна сигурност и наблюдение на здравето. Например, болногледачите, които предоставят домашни грижи за възрастни хора или хора с увреждания, могат да използват функциите за наблюдение на здравето, за да открият потенциални рискове за здравето и да предприемат действия за справяне с тях. Освен това функциите за сигурност на системата могат да осигурят допълнителен слой защита за болногледачите и техните пациенти.
4. Професионалисти по домашна сигурност: Професионалистите по домашна сигурност също могат да се възползват от модулна интернет на нещата система за домашна сигурност и мониторинг на здравето. Системата може да бъде персонализирана, за да отговаря на нуждите на различни клиенти, като предоставя набор от опции за сигурност и наблюдение на здравето. Чрез включването на най-новата интернет на нещата технология професионалистите по домашна сигурност могат да предложат на своите клиенти авангардно решение, което подобрява тяхната безопасност и благополучие.

Като цяло, едно модулно интернет на нещата приложение за домашна сигурност и мониторинг на здравето има потенциала да бъде от полза за широк кръг потребители, включително собственици на жилища, членове на семейството, лица, които се грижат за тях, и специалисти по домашна сигурност. Чрез създаването на адаптивна и свързана система от сензори и устройства собствениците на жилища могат да наблюдават показателите за сигурността на дома и личните си здравословни показатели в реално време, като получават предупреждения, когато възникнат потенциални проблеми. Резултатът е модерен и свързан дом, който подобрява безопасността, сигурността и благосъстоянието за всички участници.

# Анализ на решението

## Потребителски изисквания и работен процес

### Потребителски изисквания

Подробно описание на изискванията на потребителя за интернет на нещата система за домашна сигурност, модернизация и мониторинг на здравето:

Сигурност: Сигурността е може би най-важното изискване за интернет на нещата система за домашна сигурност и наблюдение. Потребителите искат да се чувстват сигурни и сигурни в домовете си и очакват техните системи за сигурност да осигурят стабилна защита срещу кибер заплахи и физически прониквания. Една добра система за сигурност на интернет на нещата трябва да предлага функции като криптирана комуникация между устройства, сигурно съхранение на данни и защита срещу хакване и други кибер заплахи.

Лесна употреба: Потребителите искат система, която е лесна за настройване, конфигуриране и използване. Те очакват системата да бъде интуитивна и удобна за потребителя, с ясни инструкции и достъпна онлайн поддръжка. В идеалния случай системата трябва да бъде достъпна чрез единен интерфейс, като мобилно приложение или уеб портал, който позволява на потребителите да наблюдават и контролират всички аспекти на своята домашна сигурност и система за наблюдение от едно място.

Персонализиране: Потребителите искат да могат да персонализират системата, за да отговаря на техните индивидуални нужди. Това може да включва настройване на сигнали за конкретни събития, конфигуриране на различни потребителски профили с различни нива на достъп или коригиране на настройките на системата, за да съответстват на техния начин на живот или предпочитания.

Съвместимост: Потребителите очакват тяхната интернет на нещата система за домашна сигурност и мониторинг да бъде съвместима с широк набор от устройства и платформи. Системата трябва да може да работи със смартфони, таблети и компютри, както и с други умни домашни устройства като гласови асистенти и интелигентни системи за осветление.

Интеграция: Потребителите искат тяхната интернет на нещата система за домашна сигурност и мониторинг да се интегрира безпроблемно с други интелигентни домашни устройства и услуги. Това може да включва гласови асистенти, интелигентни системи за осветление или други интернет на нещата устройства, които им позволяват да контролират различни аспекти от домашната си среда.

Модернизация: В допълнение към сигурността и наблюдението, потребителите искат техните интернет на нещата домашни системи да могат да модернизират и автоматизират различни аспекти на дома си, като контрол на осветлението и температурата, системи за забавление и потребление на енергия. Една добра интернет на нещата домашна система трябва да може да предостави на потребителите повече контрол върху домашната им среда, което я прави по-удобна и ефективна.

Мониторинг на здравето: Много потребители се интересуват от домашни системи интернет на нещата, които могат да наблюдават тяхното здраве и благосъстояние. Това може да включва проследяване на сърдечната честота или кръвното налягане, наблюдение на моделите на сън или дори проследяване на приема на храна. Потребителите очакват системата да предоставя предупреждения, когато е необходимо, като например когато конкретен здравен показател е извън здравословния диапазон.

Поверителност: И накрая, потребителите очакват техните домашни интернет на нещата системи да зачитат тяхната поверителност и поверителността на техните данни. Те искат да знаят, че личната им информация се събира, съхранява и използва отговорно и че данните им са защитени от хакери и други злонамерени участници. Добрата домашна интернет на нещата система трябва да бъде прозрачна относно своите политики за събиране и използване на данни и да предлага на потребителите набор от опции за контролиране на техните данни.

### Работен процес

Системата получава входни данни от множество източници. Първият източник са самите интернет на нещата устройства, които генерират данни, които трябва да се съхраняват в базата данни. Интернет на нещата устройствата изпращат своите данни към локалния сървър чрез TCP връзка, която се установява, когато устройството се свърже със сървъра. Данните, изпратени от устройството, включват името на устройството и колоните, необходими за съхраняване на данните. След като сървърът получи тези данни, той създава нова таблица за устройството в базата данни.

Вторият източник на входни данни е потребителският интерфейс, който се предоставя от уеб приложението и от локания сървър. Потребителите могат да взаимодействат с уеб приложението, за да преглеждат и управляват данни от интернет на нещата устройствата. Потребителският интерфейс позволява на потребителите да създават, четат, актуализират и изтриват данни от базата данни и също така им позволява да одобряват нови интернет на нещата устройства, които се опитват да се свържат със сървъра.

След като входните данни бъдат получени от системата, те преминават през фаза на обработка. Локалният сървър обработва обработката на данните, получени от интернет на нещата устройствата. Данните първо се удостоверяват, за да се гарантира, че идват от валидно устройство, което е одобрено от потребителя. Ако данните са валидни, те се съхраняват в базата данни в таблицата, която е създадена за устройството.

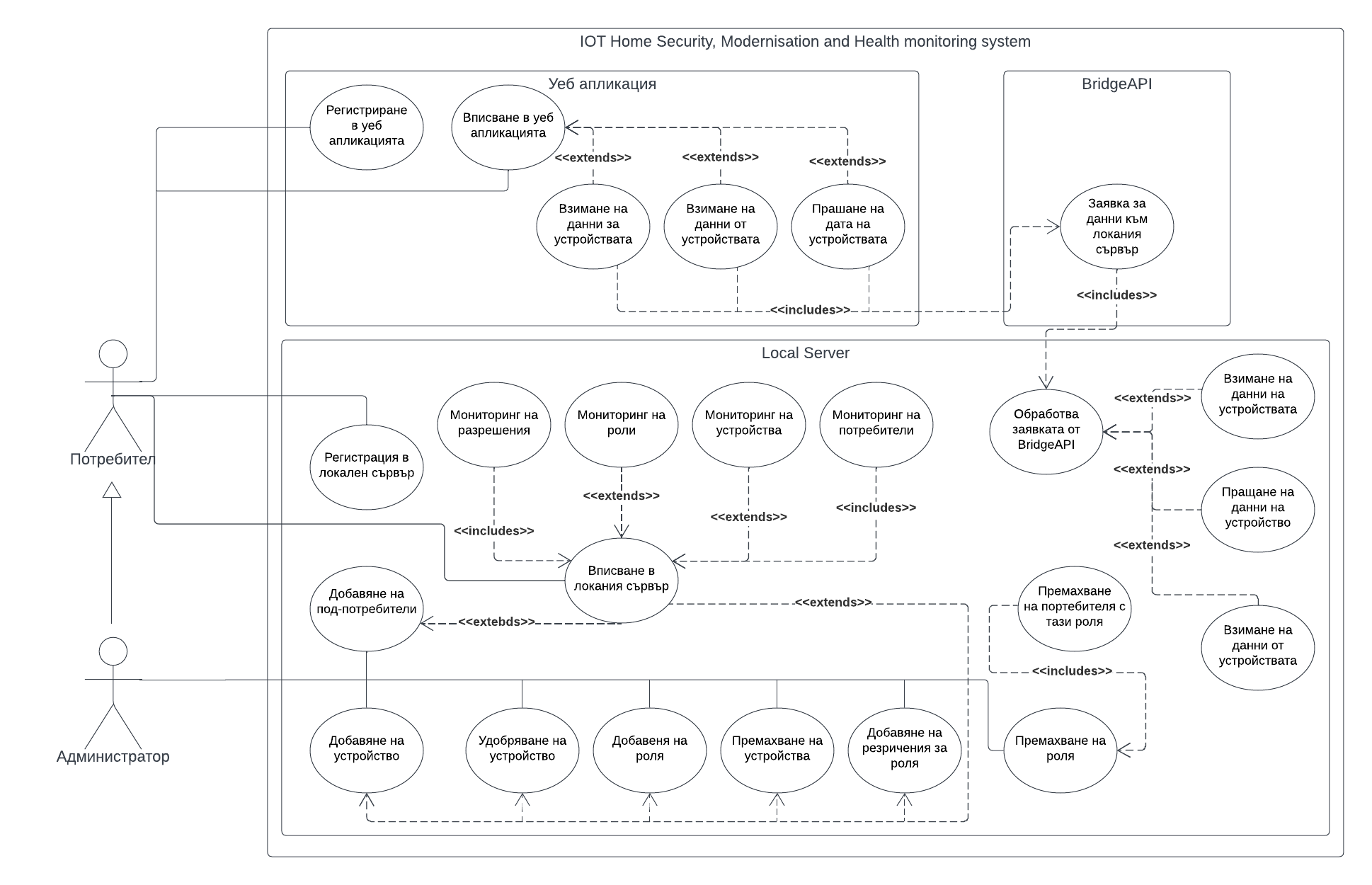
Фазата на обработка на данните, получени от уеб приложението, включва BridgeAPI, който действа като посредник между уеб приложението и локалния сървър. BridgeAPI получава заявки от уеб приложението и ги обработва, като ги препраща към локалния сървър. Локалният сървър обработва заявката и изпраща отговор обратно към BridgeAPI, който от своя страна изпраща отговора обратно към уеб приложението.

Последната фаза на системата е извеждането, което включва предоставяне на обратна връзка на потребителя въз основа на данните, съхранени в базата данни. Уеб приложението предоставя потребителски интерфейс за преглед и управление на данните. Потребителите могат да преглеждат данни в реално време или да преглеждат исторически данни, а също така могат да задават предупреждения и известия въз основа на конкретни критерии за данни.

Изходната фаза също така включва изпращане на данни обратно към интернет на нещата устройствата. Локалният сървър извлича данни от базата данни и ги изпраща обратно към устройствата чрез установената TCP връзка. След това устройствата могат да използват тези данни, за да извършват различни действия, като например управление на оборудване или предоставяне на обратна връзка в реално време на потребителя.

Като цяло работният процес на системата включва получаване на входни данни от множество източници, обработка на данните през различни етапи и предоставяне на изход въз основа на данните, съхранени в базата данни. Системата е проектирана да бъде гъвкава и мащабируема, позволявайки лесно интегриране на нови интернет на нещата устройства и разширяване на потребителския интерфейс.

#### Диаграма на случаите за използване



1. Фигура 3.1.2.1. - Диаграма на случаите

##### Пояснение по диаграмата

Регистрация на локалния сървър: Този случай на използване се отнася до процеса на регистриране на потребител на локалния сървър на системата за домашна сигурност. Актьорът „Администратор“ може да взаимодейства с този случай на употреба, като предостави своята лична информация, като име и имейл, за да се регистрира.

Регистрация в уеб приложението: Този случай на използване се отнася до процеса на регистриране на потребител в уеб приложението на системата за интернет на нещата домашна сигурност и мониторинг на здравето. Актьорът „Потребител“ може да взаимодейства с този случай на използване, като предостави своята лична информация, за да се регистрира, като име и имейл.

Добавяне на под-потребители: Този случай на използване се отнася до процеса на добавяне на под-потребители към системата за домашна сигурност. Актьорът „Администратор“ може да взаимодейства с този случай на употреба, като предостави необходимата информация за под-потребителя, като име и имейл.

Добавяне на роли: Този случай на използване се отнася до процеса на добавяне на нови роли към системата за домашна сигурност. Актьорът „Администратор“ може да взаимодейства с този случай на употреба, като създаде нова роля и дефинира нейните разрешения

Добавяне на ограничения към роля: Този случай на използване позволява на администратора да добави конкретни ограничения към роля. Например, администратор може да ограничи определена роля от достъпа до определени функции или данни в системата. Актьорът може да избере ролята, за която иска да добави ограничения и да посочи ограниченията, които иска да наложи.

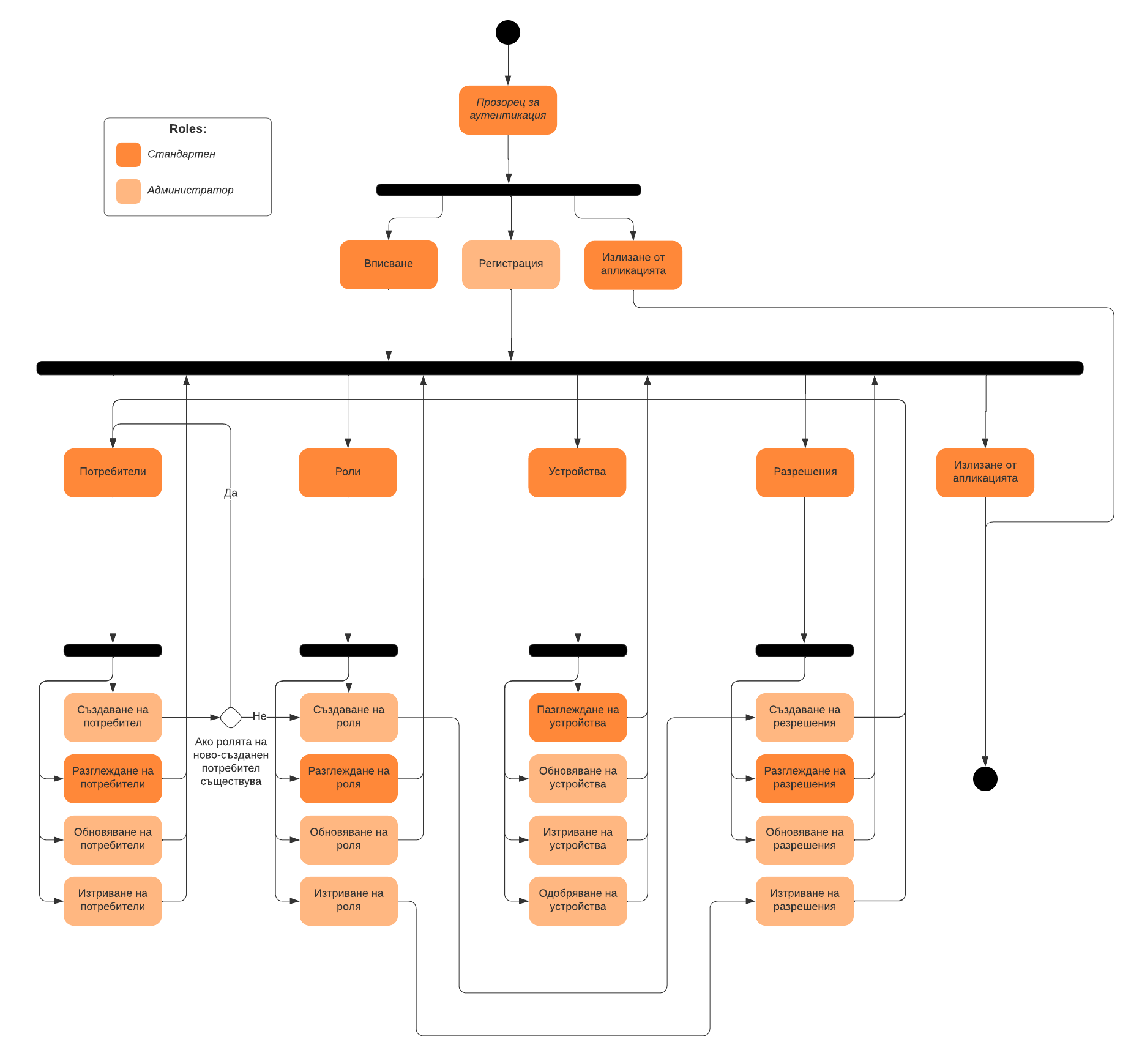
Премахване на роля: Този случай на използване позволява на администратора да премахне роля от системата. Актьорът може да избере ролята, която иска да премахне, и системата ще премахне всички разрешения, свързани с тази роля.

Влизане в локалния сървър: Този случай на използване позволява на потребителя да влезе в локалния сървър. Актьорът може да въведе своите идентификационни данни за вход и системата ще удостовери потребителя и ще му позволи достъп до функциите, налични на локалния сървър.

Влизане в уеб приложението: Този случай на използване позволява на потребителя да влезе в уеб приложението. Актьорът може да въведе своите идентификационни данни за вход и системата ще удостовери потребителя и ще му позволи достъп до функциите, налични в уеб приложението.

Получаване на данни от устройства: Този случай на използване позволява на потребителя да получава данни от своите свързани устройства. Актьорът може да избере устройството, от което иска да получи данни, и системата ще извлече данните от това устройство и ще ги покаже на потребителя. Диаграма на активността на локания сървър

#### Диаграма на активността



2. Фигура 3.1.2.2. - Диаграма на активността

##### Пояснение по диаграмата

Диаграмата на активността изобразява потока на процеса за прозорец за удостоверяване. Диаграмата започва с началното състояние (представено от символа „Начало“) и продължава към следващото състояние, което е представено от символа правоъгълник.

В системата има няколко роли, представени от правоъгълници с етикети „Стандартен“, „Администратор“ и т.н. Всяка роля има различни разрешения и действия, които могат да изпълняват, както е посочено от състоянията и преходите, които следват.

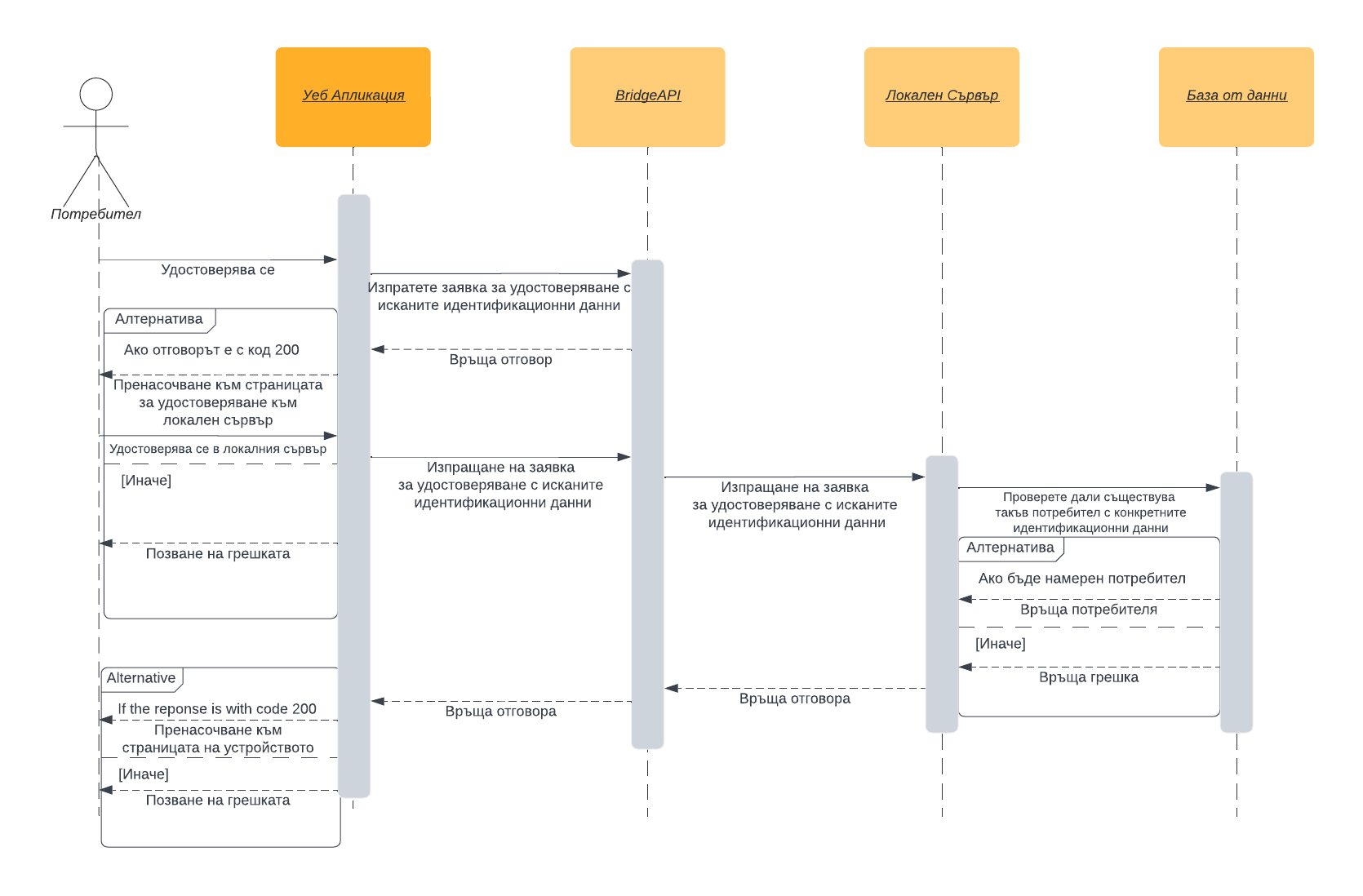
Хоризонталните символи за разклонение/съединяване показват паралелна обработка, което означава, че множество действия могат да се извършват едновременно. Например, след като влезе или се регистрира, потребителят може да продължи да разглежда списък с потребители, роли, устройства или разрешения. Всички тези действия могат да се извършват паралелно, оттук и използването на символите разклонение/съединяване.

Някои действия могат да се извършват само от определени роли. Например, действията, свързани с разрешенията (създаване, актуализиране, изтриване), могат да се извършват само от ролята „Администратор“, както е посочено от символа за решение във формата на диамант. Ако потребителят не е администратор, той няма да може да извършва тези действия.

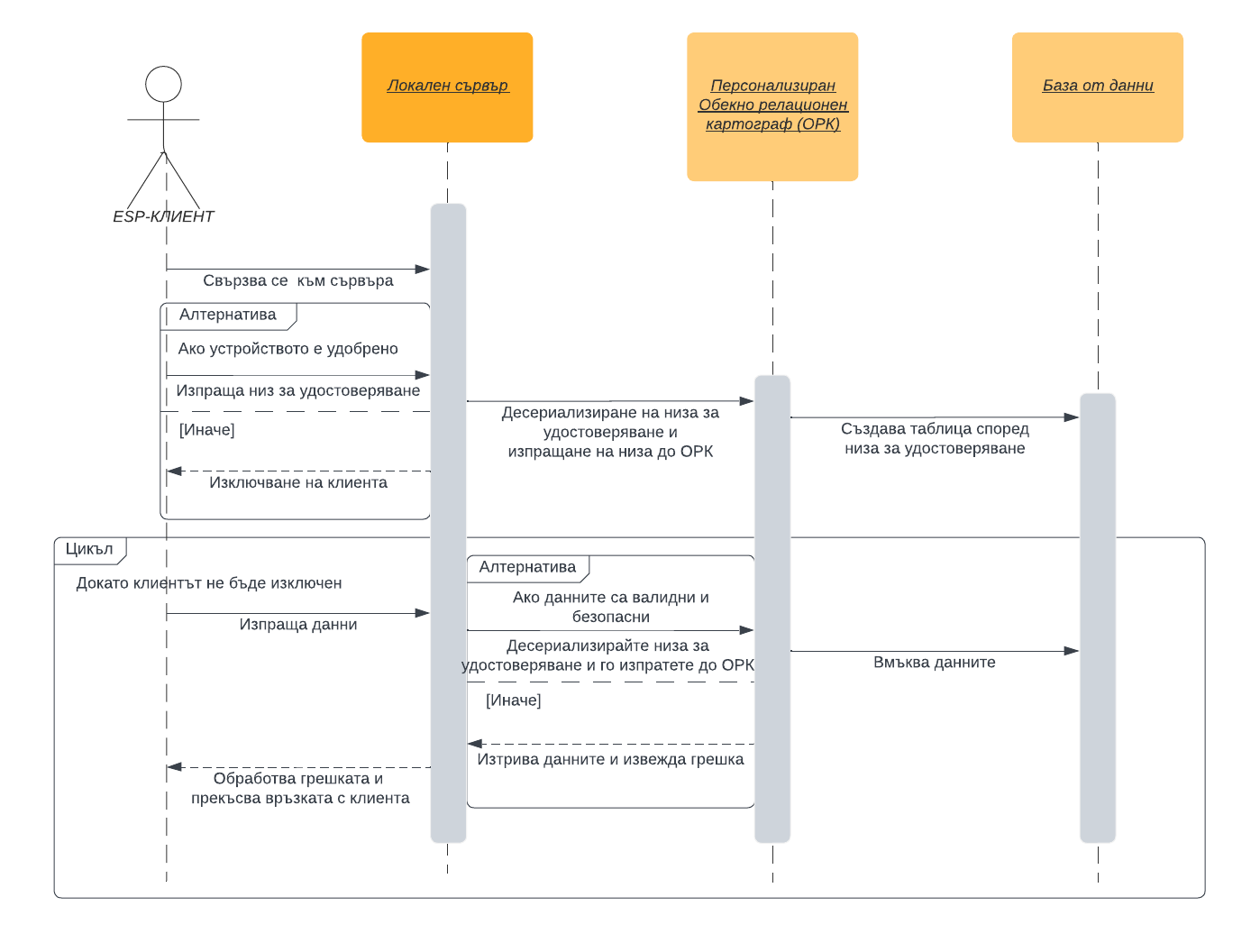
Крайните състояния на диаграмата представляват двата начина за излизане от приложението: чрез натискане на бутона "Изход" или чрез затваряне на прозореца.

Като цяло диаграмата на активността предоставя ясно и кратко представяне на потока на процеса за прозореца за удостоверяване, включително различните включени роли и разрешения.

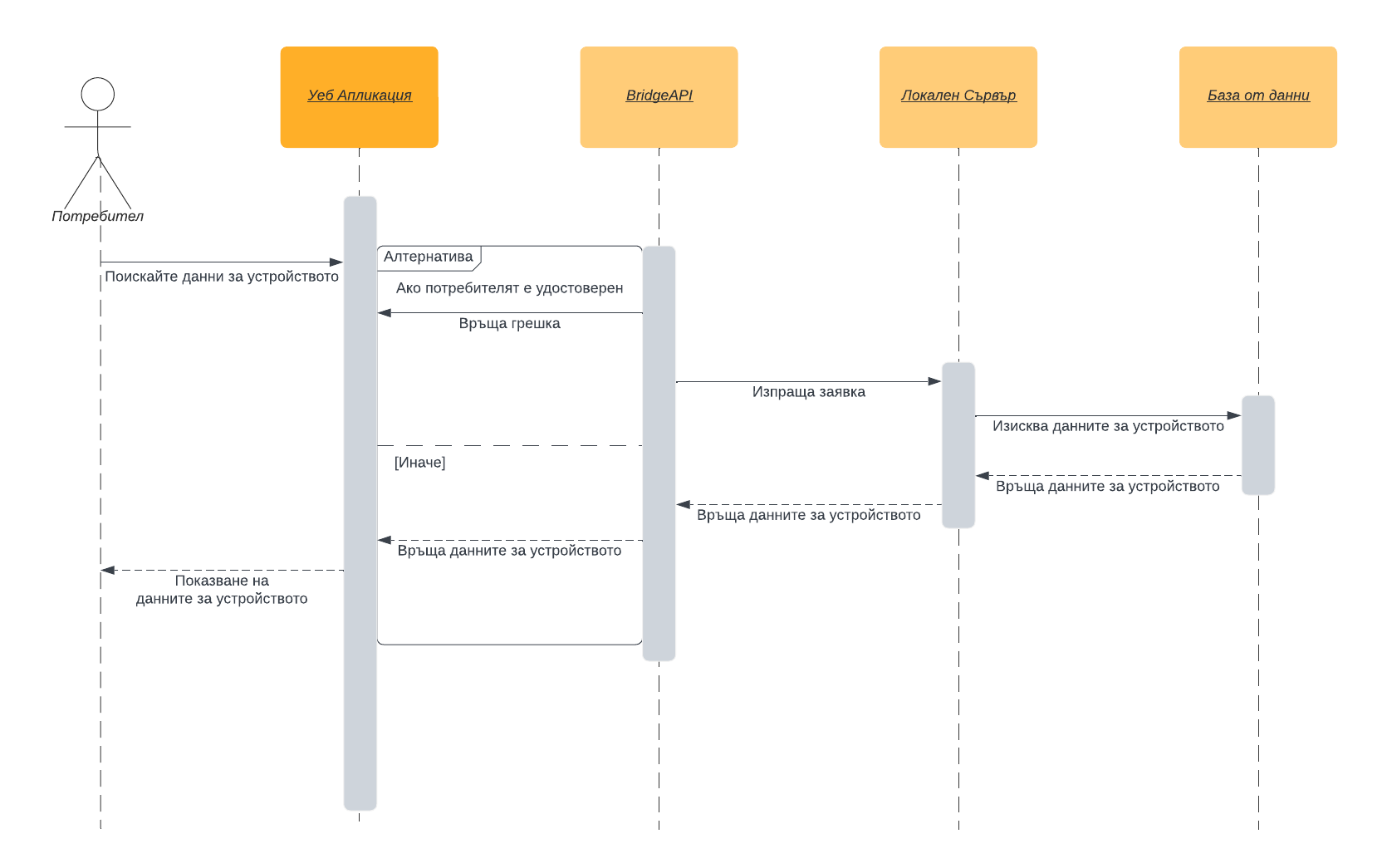
#### Диаграми на последователността



3. Фигура 3.1.2.3.1. - Диаграма на последователността за одостоверяване на потребител



4. Фигура 3.1.2.3.2. - Диаграма на последователността за връзка на интернет на нещата устройство с локалния сървър



5. Фигура 3.1.2.3.3. - Диаграма на последователността за получаване на данни от интернет на нещата устройства

##### Пояснение по диаграмите

###### Фигура 3.1.2.3.1

Процесът на удостоверяване в това приложение е решаваща стъпка за гарантиране на сигурността на системата. Процесът включва проверка на самоличността на потребителя и определяне дали потребителят има необходимите разрешения за достъп до исканите ресурси.

Когато потребител се опита да влезе в системата, той първо ще бъде подканен да въведе своите идентификационни данни за вход, които ще бъдат изпратени на сървъра за проверка. След това BridgeAPI ще провери предоставените идентификационни данни спрямо потребителска база данни, за да потвърди тяхната валидност.

Ако идентификационните данни са валидни, BridgeAPI ще генерира токен за удостоверяване, който ще бъде изпратен обратно на клиента. Този токен служи като форма на цифрова идентификация и съдържа информация за потребителя, като неговото име и разрешения.

Процесът на удостоверяване също така включва мерки за предотвратяване на неоторизиран достъп до системата. Например, сървърът може да приложи мерки като ограничаване на скоростта, за да предотврати груби атаки срещу идентификационни данни за влизане. Освен това токенът за удостоверяване може да има време на изтичане, за да ограничи прозореца на възможност за неоторизиран достъп.

Когато потребител се опита да влезе в локалния сървър чрез уеб приложението, заявката за влизане първо се изпраща до BridgeAPI. BridgeAPI действа като посредник между уеб приложението и локалния сървър и отговаря за обработката на удостоверяване и оторизация.

При получаване на заявката за влизане, BridgeAPI първо проверява дали потребителят вече е удостоверен, като търси идентификатор на сесия в бисквитките на браузъра на потребителя. Ако бъде намерен идентификатор на сесия, потребителят се счита за удостоверен и BridgeAPI позволява заявката за влизане да премине към локалния сървър.

Ако не бъде намерен идентификатор на сесия, BridgeAPI подканва потребителя да въведе своите идентификационни данни за вход, като потребителско име и парола. След това BridgeAPI изпраща идентификационните данни за вход към локалния сървър за удостоверяване.

В локалния сървър идентификационните данни за вход се проверяват спрямо идентификационните данни, съхранени в базата данни. Ако идентификационните данни са валидни, локалният сървър генерира нов идентификатор на сесия за потребителя и го съхранява в базата данни. След това идентификационният номер на сесията се връща към BridgeAPI, който задава нова бисквитка за сесия в браузъра на потребителя и позволява на заявката за влизане да премине към локалния сървър.

Ако идентификационните данни за вход са невалидни, локалният сървър връща съобщение за грешка на BridgeAPI, който от своя страна връща съобщение за грешка на уеб приложението, което показва, че влизането е неуспешно.

###### Фигура 3.1.2.3.3

Когато се направи заявка за данни за устройството от уеб приложението, заявката не се изпраща директно до локалния сървър. Вместо това преминава през BridgeAPI, който действа като посредник между уеб приложението и локалния сървър. Този процес на препредаване на заявката гарантира, че само удостоверени и оторизирани заявки се предават на локалния сървър.

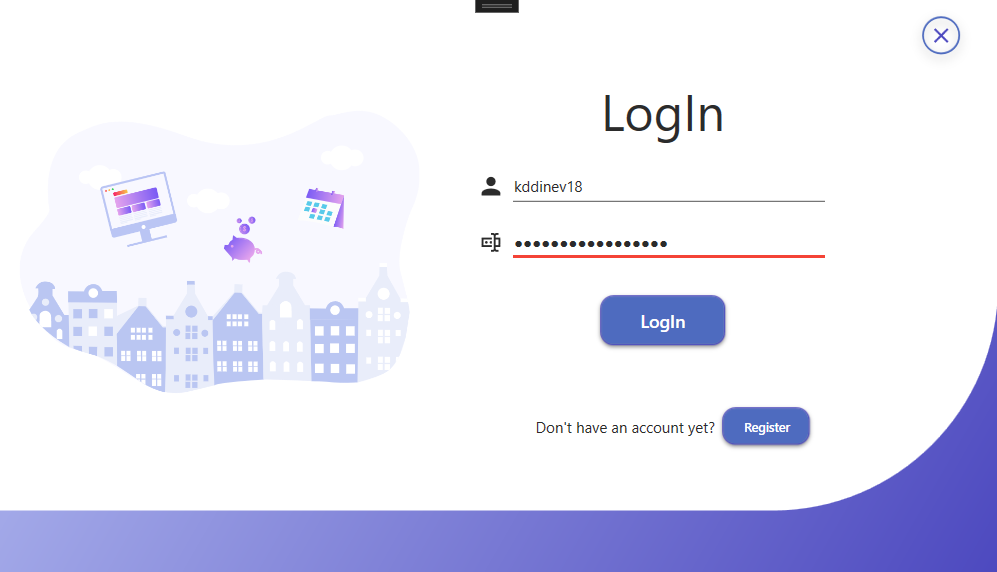
BridgeAPI моста проверява удостоверяването и оторизацията на потребителя, който прави заявката, преди да я препрати към локалния сървър. Това се прави, за да се гарантира, че само оторизирани потребители имат достъп до данните на локалния сървър. API на моста проверява идентификационните данни на потребителя спрямо база данни от оторизирани потребители и продължава със заявката само ако потребителят е упълномощен. Ако потребителят не е упълномощен, мостовият API ще върне съобщение за грешка на уеб приложението.

След като BridgeAPI потвърди, че потребителят е оторизиран, той препраща заявката към локалния сървър. Локалният сървър получава заявката и я обработва, за да извлече исканите данни от базата данни. След това локалният сървър изпраща извлечените данни обратно към BridgeAPI, който от своя страна ги предава на уеб приложението.

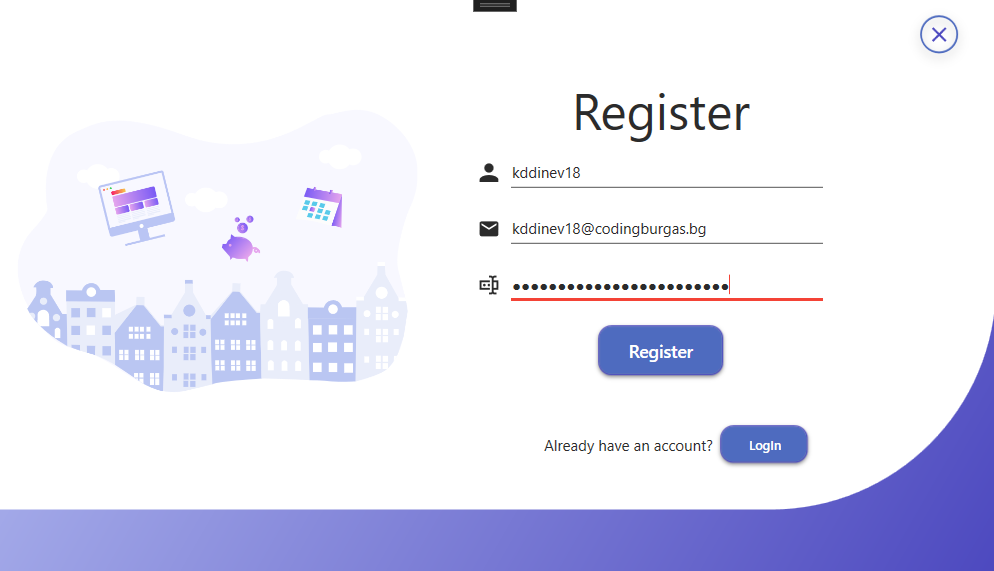
По време на процеса на препредаване на заявката и данните между уеб приложението и локалния сървър, BridgeAPI гарантира, че данните се предават сигурно. BridgeAPI използва алгоритми за криптиране, за да защити данните от неоторизиран достъп и прихващане по време на предаване.

## Примерен потребителски интерфейс

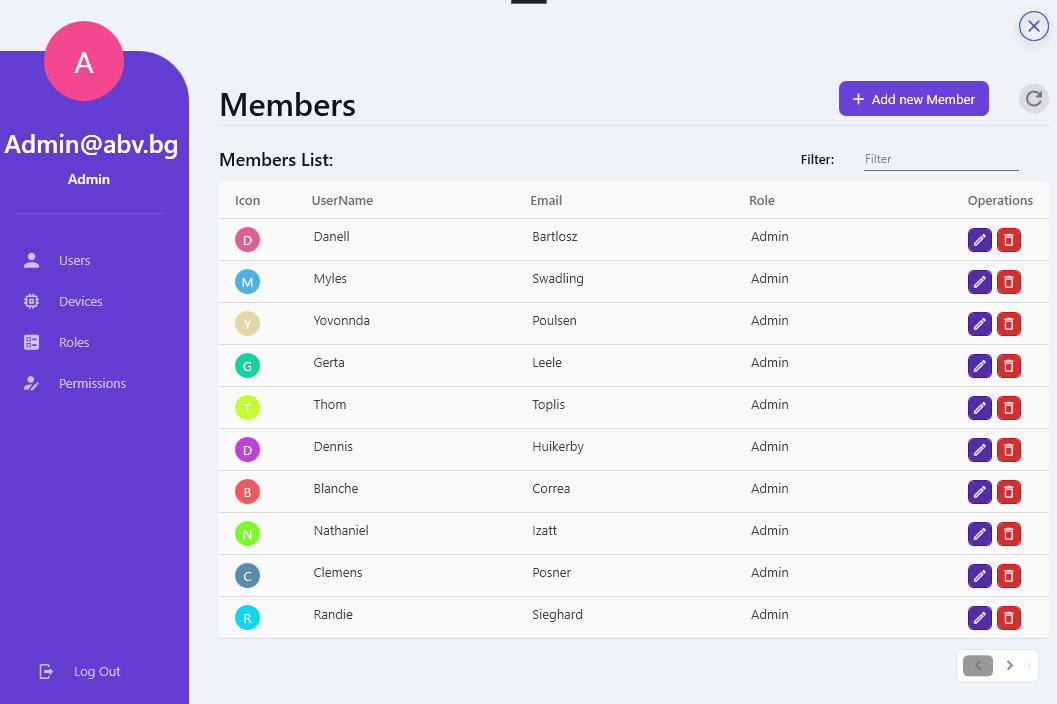
### Локален сървър



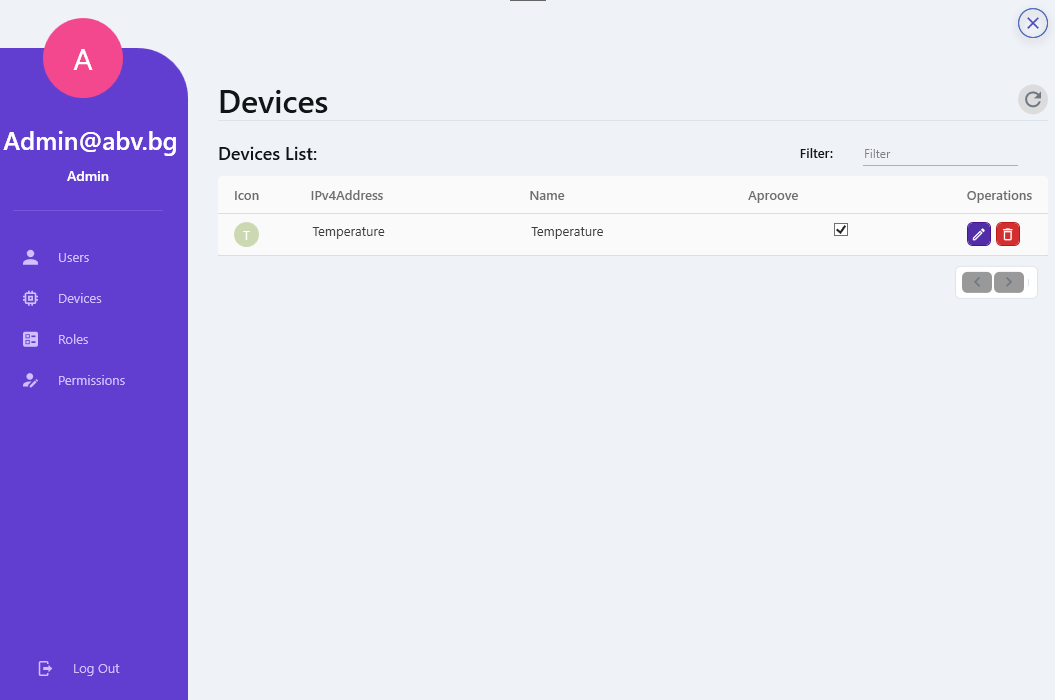
6. Фигура 3.2.1.1. - Дизайн на локалния сървър



7. Фигура 3.2.1.2. - Дизайн на локалния сървър



8. Фигура 3.2.1.3. - Дизайн на локалния сървър



9. Фигура 3.2.1.4. - Дизайн на локалния сървър

## Диаграми на анализа

## Диаграма на връзката на обекта

### C:\Users\kddinev18\Downloads\ERD import example.png Локален Сървър

11. Фигура 3.4.1.- Диаграма на връзките на обекта

#### Описание по диаграмата

Базата данни е релационна база данни, която съхранява информация за интернет на нещата устройства, потребители, роли и разрешения, както и данните от сензорите, събрани от тези устройства.

Таблицата за устройства (Devices) съхранява информация за всяко устройство в системата, включително неговия уникален идентификатор (Id), IP адрес (IPv4Address), име (Name) и дали е одобрено за използване (IsApproved). Тази таблица позволява на системата да следи всички устройства в системата и техните свойства.

Таблицата за роли (Roles) определя различните роли, които потребителите могат да имат в системата. Например, може да има роля „администратор“ с пълен достъп до всички функции и роля „потребител“ с по-ограничен достъп. На всяка роля се присвоява уникален идентификатор (Id) и име (Name), което позволява на системата да следи различните налични роли.

Таблицата с разрешения(Permissions) съхранява информация за това кои роли имат какви разрешения за всяко устройство. Това позволява на системния администратор да контролира кои потребители какви действия могат да извършват на всяко устройство. Например администратор може да има пълен достъп до всички устройства, докато потребител може да има достъп само за четене до определени устройства. Всеки запис в таблицата с разрешения указва роля (RoleId), устройство (DeviceId) и дали тази роля има разрешение да създава, чете, актуализира или изтрива данни за това устройство.

Таблицата потребители(Users) съхранява информация за всеки потребител в системата, включително уникален идентификатор (Id), потребителско име (UserName), имейл адрес (Email), парола (Password), солна стойност (Salt) за криптиране на паролата, и ролята на потребителя (RoleId). Тази таблица позволява на системата да следи всички потребители и техните идентификационни данни.

И накрая, за всеки тип сензорни данни, които системата събира, има отделна таблица (като температурната таблица, която описахме). Тези таблици съдържат колони за данните на сензора (като температура и влажност), както и уникален идентификатор (Id), клеймо за време, указващо кога данните са били събрани (Създаден), и уникалния идентификатор на устройството(DeviceId), което е събрало данните. Това позволява на системата да следи всички сензорни данни, събрани от различните устройства в системата.

#### Нормализация на базата от данни

Устройства(Devices): Тази таблица има един първичен ключ (Id) и няма повтарящи се групи или вложени таблици, така че отговаря на изискванията за първа нормална форма. Освен това всички неключови атрибути (IPv4Address, Name, IsApproved) са функционално зависими от първичния ключ, което означава, че всеки атрибут се определя еднозначно от първичния ключ(DeviceId). Следователно отговаря на изискванията за втора нормална форма. И накрая, няма транзитивни зависимости между неключови атрибути, което означава, че няма неключови атрибути, които зависят от други неключови атрибути. Следователно отговаря на изискванията за трета нормална форма.

Роли(Roles): Тази таблица има един първичен ключ (Id) и няма повтарящи се групи или вложени таблици, така че отговаря на изискванията за първа нормална форма. Освен това има само един неключов атрибут: име (Name), който е функционално зависим от първичния ключ (Id). Следователно отговаря на изискванията за втора нормална форма. И накрая, няма транзитивни зависимости между неключови атрибути, което означава, че няма неключови атрибути, които зависят от други неключови атрибути. Следователно отговаря на изискванията за трета нормална форма.

Разрешения(Permissions): Тази таблица има съставен първичен ключ, състоящ се от два външни ключа (RoleId и DeviceId), което позволява да се присвоят множество разрешения за всяка комбинация роля/устройство. Няма повтарящи се групи или вложени таблици, така че отговаря на изискванията за първа нормална форма. Освен това всички неключови атрибути (CanCreate, CanRead, CanUpdate, CanDelete) са функционално зависими от съставния първичен ключ, което означава, че всеки атрибут се определя уникално от комбинацията от съставния първичен ключ (RoleId и DeviceId). Следователно отговаря на изискванията за втора нормална форма. И накрая, няма транзитивни зависимости между неключови атрибути, което означава, че няма неключови атрибути, които зависят от други неключови атрибути. Следователно отговаря на изискванията за трета нормална форма.

Потребители(Users): Тази таблица има един първичен ключ (Id) и няма повтарящи се групи или вложени таблици, така че отговаря на изискванията за първа нормална форма. Освен това всички неключови атрибути (UserName, Email, Password, Salt, RoleId) са функционално зависими от първичния ключ, което означава, че всеки атрибут се определя еднозначно от първичния ключ (Id). Следователно отговаря на изискванията за втора нормална форма. И накрая, няма транзитивни зависимости между неключови атрибути, което означава, че няма неключови атрибути, които зависят от други неключови атрибути. Следователно отговаря на изискванията за трета нормална форма.

Температура (и други таблици с данни от сензори): Всяка от тези таблици има един първичен ключ (Id) и няма повтарящи се групи или вложени таблици, така че отговаря на изискванията за първа нормална форма. Освен това, всички неключови атрибути (Temperature, Humididty, Created, DeviceId) са функционално зависими от първичния ключ, което означава, че всеки атрибут се определя еднозначно от идентификатора. Следователно отговаря на изискванията за втора нормална форма. И накрая, няма транзитивни зависимости между неключови атрибути, което означава, че няма неключови атрибути, които зависят от други неключови атрибути. Следователно отговаря на изискванията за трета нормална форма.

#### Описание на връзките

Връзките в базата данни могат да бъдат описани като начина, по който таблиците са свързани една с друга.

Започвайки с таблицата за устройства (Devices), можем да видим, че тя има три основни връзки с други таблици в базата данни. Първо, едно устройство може да има много разрешения (Permissions), свързани с него, които са определени в таблицата с разрешения. Всяко разрешение е свързано с конкретно устройство и роля и дефинира нивото на достъп, което потребител с тази роля има за това устройство. Например, потребител с роля „Администратор“ може да има пълен достъп до устройство, докато потребител с роля „Гост“ може да може само да преглежда състоянието на устройството.

Второ, едно устройство може да има много температурни показания, свързани с него, които се съхраняват в температурната таблица. Това предполага, че устройството е температурен сензор, който може периодично да изпраща данни за температура и влажност на приложението. Всяко отчитане на температурата е свързано с конкретно устройство и се съхранява с клеймо за време, показващо кога е взето отчитането.

И накрая, едно устройство може да бъде свързано само с един потребител, който е дефиниран в таблицата потребители (Users). Това позволява на всеки потребител да има свой собствен набор от устройства, които може да управлява, наблюдава и контролира.

Преминавайки към таблицата Роли (Roles), виждаме, че тя е свързана с таблиците разрешения (Premissions) и потребители (Users). Ролята е колекция от разрешения, които определят какви действия е разрешено да извършва потребител с тази роля. Например, на потребител с роля „Администратор“ може да бъде разрешено да добавя, променя или изтрива устройства, докато на потребител с роля „Гост“ може да бъде разрешено само да преглежда състоянието на устройството.

Ролите се свързват с разрешения чрез таблицата с разрешения, която има връзка едно към много между роли и устройства. Това позволява гъвкавост при присвояването на роли на устройства, тъй като едно устройство може да има една роля, свързана с него, и една роля може да бъде свързана с много устройства.

И накрая, таблицата потребители (Users) е свързана с таблицата устройства (Devices) чрез връзка едно към много. Това позволява на всеки потребител да има множество устройства, свързани с неговия акаунт, и позволява лесно управление на устройствата от потребителя.

## Модел на съдържанието / данните

Съдържанието на едно приложение може да се дефинира широко като информацията или данните, които се представят на потребителя. В случая на описаното приложение за мониторинг, съдържанието ще бъде данните в реално време, получени от сензорите и устройствата, свързани към системата. Тези данни ще включват набор от показатели като температура, влажност, налягане и други параметри на околната среда, както и всякакви други данни, които са от значение за конкретния случай на използване на приложението.

Едно от ключовите предизвикателства при проектирането на ефективен модел на съдържание за този тип приложения е да се гарантира, че данните са представени по начин, който е едновременно информативен и приложим. За да се постигне това, е важно да се вземат предвид нуждите и очакванията на потребителите, както и конкретните случаи на употреба, които приложението е проектирано да поддържа.

Друго важно съображение при проектирането на модела на съдържание за интернет на нещата приложение е необходимостта от поддръжка на визуализация и анализ на данни в реално време. Това изисква гъвкава и мащабируема архитектура на данни, която може да поддържа обработката и анализа на големи обеми данни в реално време.

По отношение на формата на данните, съдържанието на интернет на нещата приложение са структурирани данни, като JSON. Конкретният използван модел на данни ще зависи от конкретния случай на употреба и типовете данни, които се събират и анализират. Дизайн

# Дизайн

## Реализация на архитектурата на приложението

Системата за интернет на нещата за домашна сигурност, модернизация и мониторинг на здравето използва много технологии, най-тривиалните и най-главните от, които са:

SQL Server: SQL Server е система за управление на релационни бази данни, която съхранява и управлява данни ефективно. Използва се в тази система за съхраняване на данните, събрани от Интернет на нещата устройствата, включително показания на сензори, изображения и видеоклипове. SQL Server е известен със своята скалируемост и висока производителност, което го прави чудесен избор за работа с големи количества данни. Той също така предоставя разширени функции за сигурност, като криптиране на данни и контрол на достъпа, които са от съществено значение за осигуряването на чувствителни данни.

WPF с XAML и C#: WPF (Windows Presentation Foundation) е графична подсистема в Windows за изобразяване на потребителски интерфейси. Той предоставя унифицирана рамка за изграждане на десктоп приложения, които имат богат потребителски интерфейс. В тази система WPF се използва с XAML (Extensible Application Markup Language) за създаване на предния край на приложението на локалния сървър. Кода зад графичния потребителски интерфейс и логиката на локалния сървър са написани на C#, модерен и широко използван език за програмиране, който е известен със своята производителност, надеждност и лекота на използване.

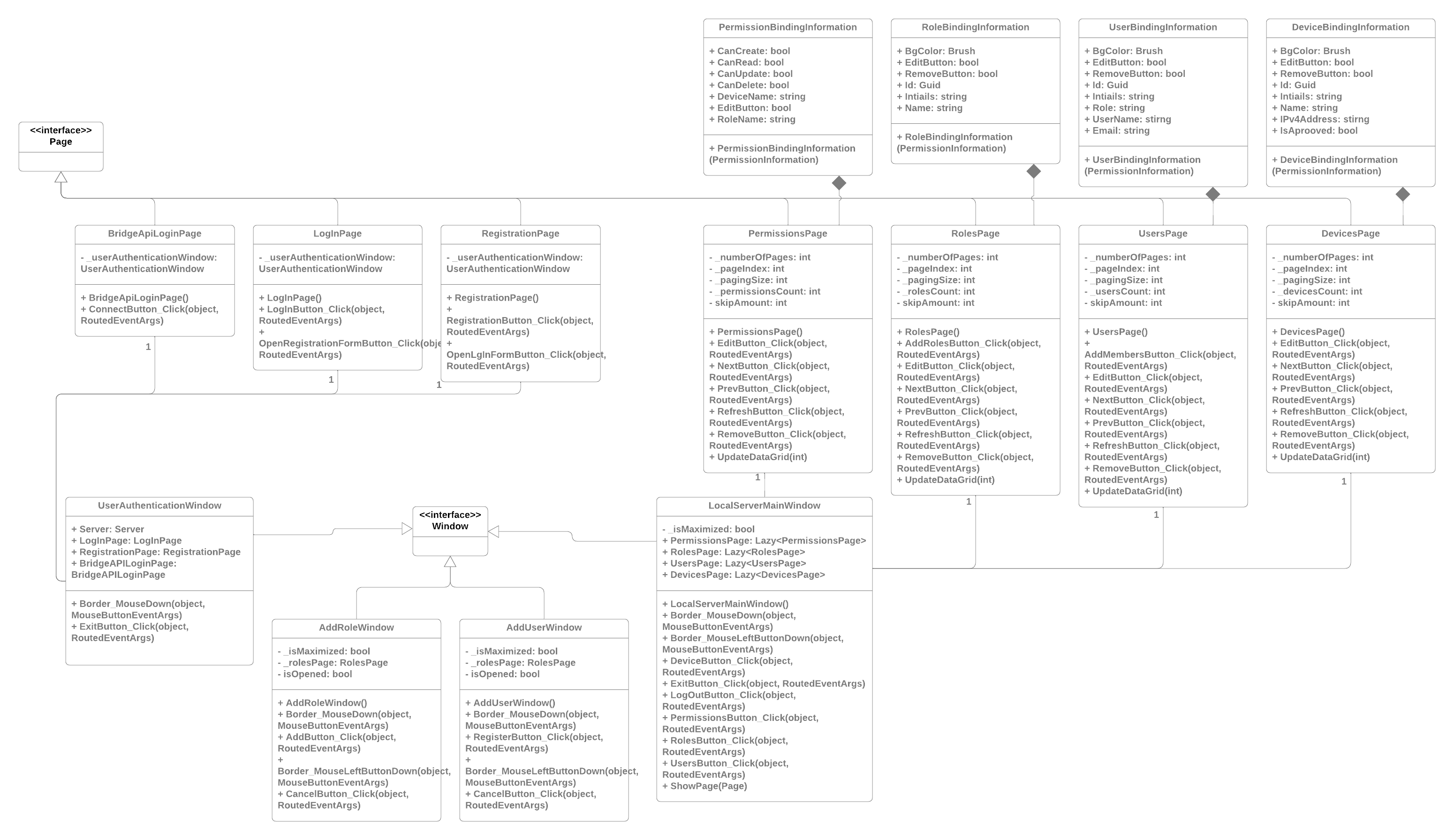
ASP.NET Core MVC: ASP.NET Core е уеб рамка с отворен код за изграждане на модерни уеб приложения. Той е бърз, междуплатформен и поддържа широка гама от езици за програмиране. В тази система ASP.NET Core се използва за разработване както на BridgeAPI, така и на уеб приложението. BridgeAPI предоставя стандартизиран интерфейс за достъп и манипулиране на данни от Интернет на нещата устройства. Уеб приложението се използва от потребителя за наблюдение и контрол на устройствата. Архитектурният модел MVC (Model-View-Controller) се използва в ASP.NET Core, което насърчава разделянето на проблемите и модулността.

Като цяло технологиите, използвани в тази система, са избрани заради тяхната производителност, надеждност, сигурност и лекота на използване. Всяка технология играе критична роля за гарантиране, че системата функционира гладко и осигурява безпроблемно потребителско изживяване.

## Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

### Диаграми на класовете на локалния сървър

#### Диаграма на класовете на графичния потребителски интерфейс на локалния сър



12. Фигура 4.2.1.1. - Диаграма на класовете на графичния потребителски интерфейс на класовете на локалния сървър

##### Описание по диаграмата

Диаграма описва класовете на графичния потребителски интерфейс. Има три вида класове, описани в диаграмата.

Класове за прозорци (Window): Класовете, които са зад кода на XAML Window, осигуряват логиката и поведението за графичния потребителски интерфейс (GUI), дефиниран в съответния XAML файл. Тези класове обикновено произлизат от класа Window в .NET Core и съдържат методи и свойства, които позволяват взаимодействие между графичния потребителски интерфейс (GUI) и основната бизнес логика на приложението.

Някои често срещани методи, намиращи се в класовете зад кода, включват манипулатори на събития за въвеждане от потребителя, методи за обвързване на данни и валидиране и логика за навигация. Обработчиците на събития се използват, за да отговорят на въвеждане от потребителя, като щракване върху бутон или избор на меню. Методите за обвързване на данни се използват за обвързване на данни от модела на данни на приложението към елементи в графичния потребителски интерфейс (GUI). Методите за валидиране се използват, за да се гарантира, че въведеното от потребителя отговаря на зададените критерии и за предотвратяване на повреда на данните. Навигационната логика се използва за превключване между различни изгледи или страници в приложението.

В допълнение към тези общи методи, класовете зад кода могат също да съдържат свойства, които определят поведението на графичния потребителски интерфейс (GUI). Тези свойства могат да контролират различни аспекти на графичния потребителски интерфейс (GUI), като неговия външен вид, оформление и взаимодействие. Например, свойства като височина, ширина, фон и IsEnabled могат да се използват за промяна на размера, цвета и наличността на графичния потребителски интерфейс (GUI) елементи.

Освен това, кодираните класове могат също да взаимодействат с други класове в приложението. Тези класове могат да бъдат класове за достъп до данни, класове на бизнес логика или други класове на графичния потребителски интерфейс (GUI). Например, клас зад код за прозорец за управление на клиенти може да взаимодейства с клас за достъп до база данни на клиенти, за да извлече и модифицира клиентски данни. Като алтернатива, той може да взаимодейства с клас бизнес логика, който извършва изчисления или прилага бизнес правила към данните.

Класове за страници (Page): В WPF (Windows Presentation Foundation) страниците се използват за представяне на различни изгледи или екрани в приложение. Те обикновено се дефинират в XAML (разширяем език за маркиране на приложения) и поведението им се определя от съответния код зад класове, които са написани на C# или други .NET езици.

Всеки клас страница в WPF е отговорен за обработката на въведеното от потребителя, обработката на данни и съответното актуализиране на потребителския интерфейс. Те могат също така да взаимодействат с други класове, като модели на изгледи или класове услуги, за да извличат или манипулират данни, ако е необходимо.

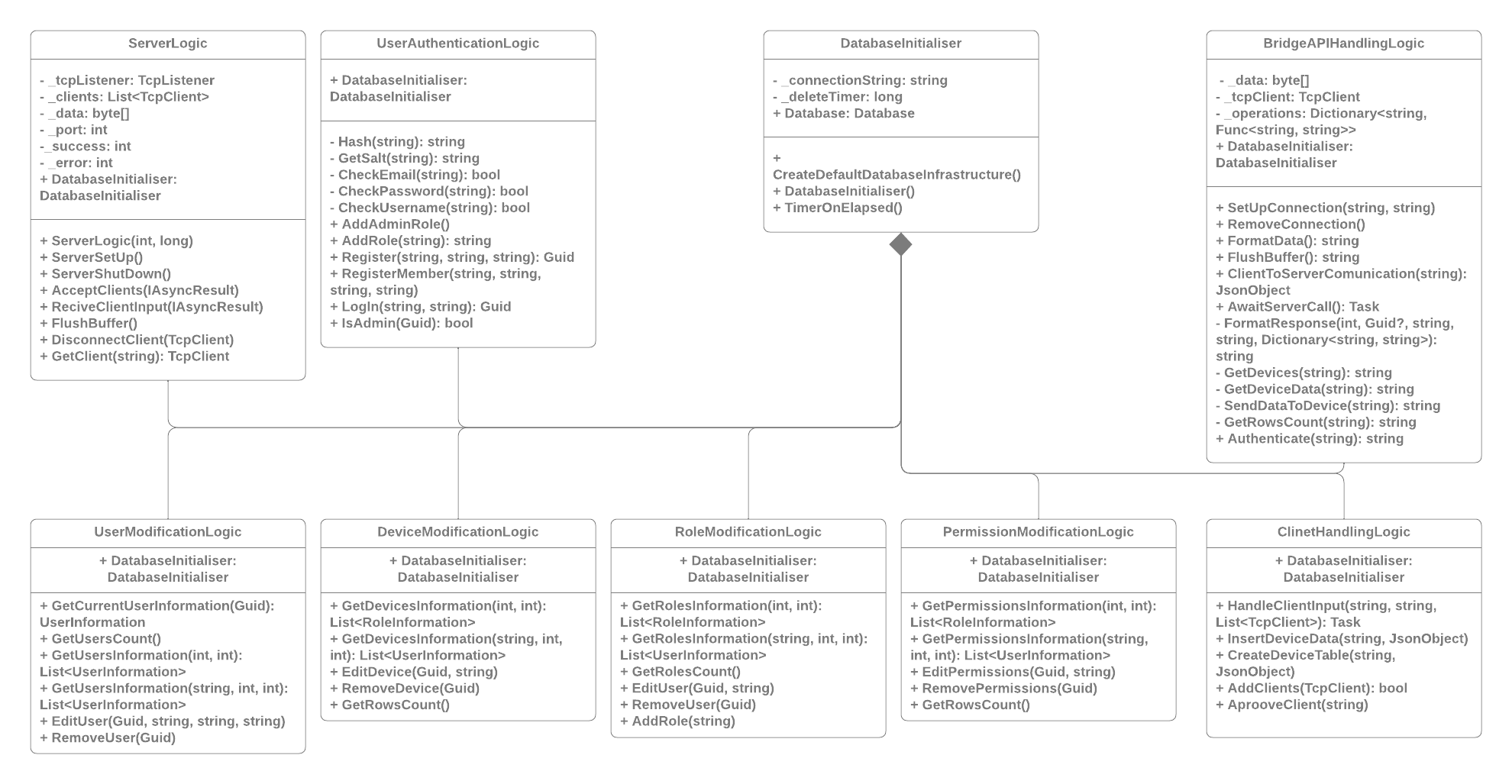
1. Класовете страници в WPF са извлечени от класа System.Windows.Controls.Page и обикновено включват следните членове:
2. XAML маркиране: Това дефинира структурата, оформлението и външния вид на страницата, включително всички елементи на потребителския интерфейс като бутони, текстови полета и мрежи с данни.
3. Обработчици на събития: Това са методи, които обработват събития, въведени от потребителя, като щраквания върху бутони, въвеждане на текст или промени в селекцията в мрежа с данни.
4. Обвързване на данни: Те определят връзките между елементите на потребителския интерфейс и данните, които показват или манипулират. Те могат да бъдат дефинирани в XAML или в задния код.
5. Свойства и полета: Те съхраняват данни, които са специфични за страницата, като текущата селекция в мрежа с данни, текстът, въведен в текстово поле, или резултатите от заявка за данни.
6. Методи: Те изпълняват различни задачи, свързани със страницата, като например актуализиране на потребителския интерфейс в отговор на въвеждане от потребителя, извличане на данни от услуга или валидиране на въведеното от потребителя.

Класове за обвързване на информация (Binding Information): В WPF (Windows Presentation Foundation) обвързването на данни е основна функция, която позволява данните да се показват и актуализират динамично в потребителски интерфейс. Най-често използваната структура на данни в WPF е наблюдаема колекция, която е специализирана колекция, която позволява обектите да бъдат автоматично добавяни, премахвани и актуализирани, когато се променят в основния източник на данни. Наблюдаемите колекции са особено полезни при обвързване на данни с контроли на потребителския интерфейс, като списъци, таблици и мрежи, тъй като те предоставят начин за поддържане на потребителския интерфейс в синхрон с източника на данни без необходимост от сложен код за обработка на събития.

Наблюдаваната колекция е по същество обвивка около стандартна колекция, като например списък или масив, която предизвиква събития всеки път, когато елементи се добавят, премахват или актуализират. Това позволява на потребителския интерфейс да се абонира за тези събития и автоматично да се актуализира в отговор на промени в източника на данни. Например, ако потребител добави нов елемент към списък, който е обвързан с наблюдаема колекция, колекцията ще предизвика събитие, уведомяващо потребителския интерфейс за промяната, и потребителският интерфейс ще се актуализира, за да покаже новия елемент. По същия начин, ако даден елемент бъде премахнат от колекцията, потребителският интерфейс ще премахне съответния елемент от дисплея.

За да използвате наблюдаеми колекции в WPF, трябва да дефинирате клас Plain Old CLR Object (POCO), който да представя всеки елемент в колекцията. POCO класовете са прости класове, които съдържат свойства, които представляват данните, които искате да свържете към потребителския интерфейс. Например, ако създавате списък с клиенти, вашият POCO клас може да съдържа свойства като име, адрес и телефонен номер. След като сте дефинирали вашия POCO клас, можете да създадете наблюдаема колекция от този тип клас и да я свържете към контролата на потребителския интерфейс.

#### Диаграма на класовете на бизнес логиката на локалния сървър



13. Фигура 4.2.1.2. - Диаграма на класовете на бизнес логиката на локалния сървър

##### Описание по диаграмата

Бизнес логиката на локания сървър се състои от три главни компонента:

Логика за сървъра: Класът ServerLogic е критичен компонент на локалното сървърно приложение, който осигурява необходимата логика за обработка на входящи данни от интернет на нещата устройства.

Един важен аспект на класа ServerLogic е способността му да работи с множество устройства едновременно. Тъй като сървърът е проектиран да обработва множество интернет на нещата устройства, класът ServerLogic е създаден да обработва данни от различни източници едновременно. Тази функция гарантира, че данните се получават в реално време и че няма изоставено устройство.

Друг важен аспект на класа ServerLogic е способността му да удостоверява входящите данни. Всяко устройство, което се свързва с локалния сървър, трябва да има уникален ключ за удостоверяване, за да се гарантира, че е упълномощено да изпраща данни. Класът ServerLogic управлява този процес на удостоверяване и гарантира, че само удостоверени устройства имат право да изпращат данни към сървъра.

В допълнение към автентификацията, класът ServerLogic е отговорен и за анализирането на входящите данни и преобразуването им във формат, който може лесно да се съхранява и анализира. Класът прилага бизнес правила към данните, за да гарантира тяхната точност и последователност. След като данните бъдат анализирани, те се съхраняват в базата данни с помощта на класа за персонализирана обектно релационна картографна (ORM) система, който е персонализиран инструмент за обектно-релационно картографиране, предназначен да работи със специфичните нужди на локалното сървърно приложение.

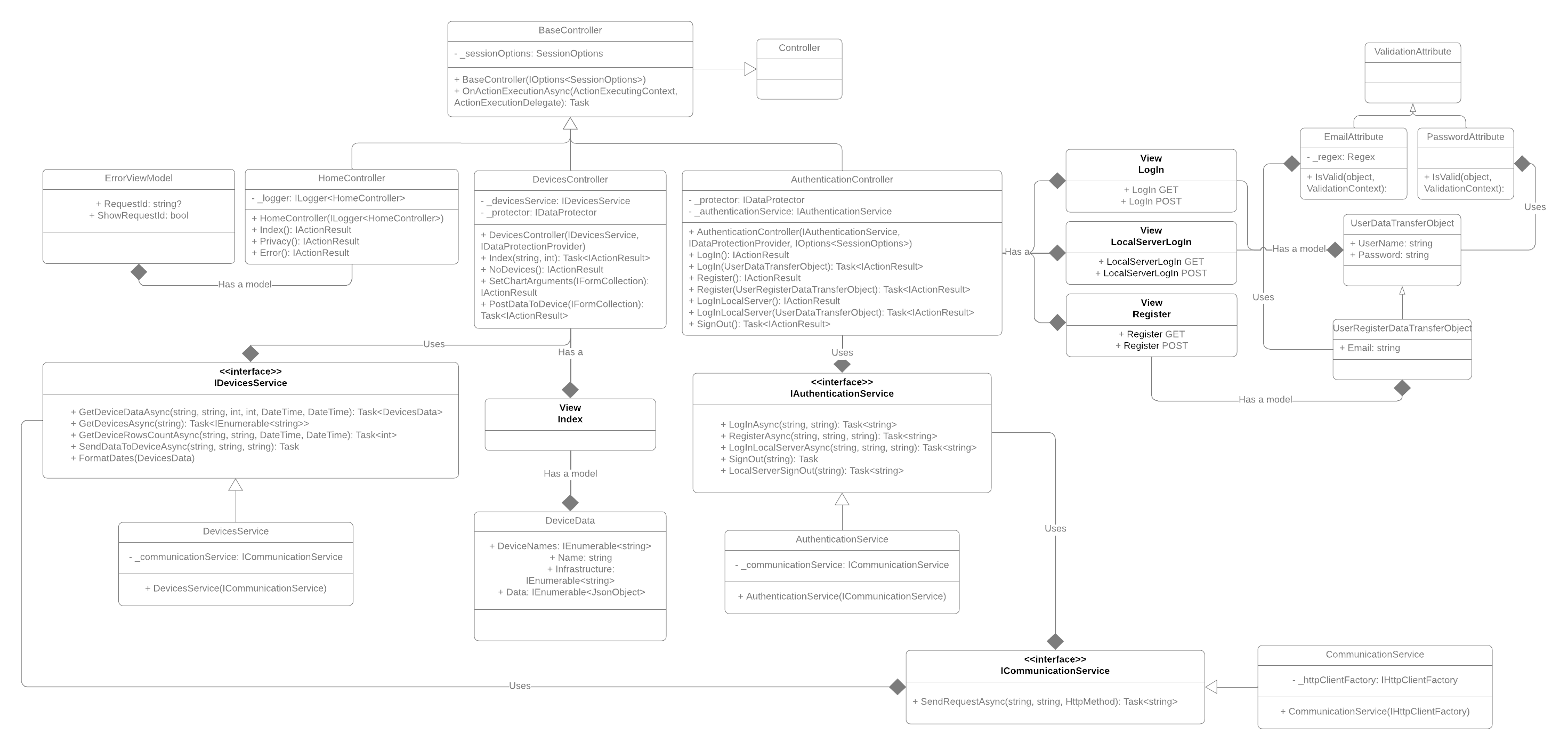
Класът ServerLogic също включва функционалност за обработка на грешки. Ако възникне грешка по време на обработката на данни, класът ServerLogic може да генерира съобщение за грешка и да обработи грешката по подходящ начин. Това гарантира, че сървърът остава стабилен и че данните не се губят поради неочаквани грешки.

Логика за манипулация на данните: Класовете за манипулиране на данни са отговорни за модифицирането на данните в базата данни. Те използват персонализиранa обектно релационна картографна (ORM) система за взаимодействие с базата данни, а не LINQ. Тези класове обработват операции като добавяне, актуализиране или изтриване на данни от базата данни. Те са проектирани да работят с различни таблици в базата данни, като например таблици, които съхраняват информация за потребители, роли, разрешения и интернет на нещата устройства. Персонализираният обектно релационен картограф (ORM), използван от класовете, абстрахира детайлите на базата данни, което улеснява работата с данните, без да се налага да пишете SQL заявки на ниско ниво. Този подход позволява по-голяма гъвкавост при работа с базата данни, както и по-лесна поддръжка и мащабируемост на приложението.

Логика за комуникация с BridgeAPI: BridgeAPIHandlingLogic е ключов компонент в процеса на комуникация между уеб приложението и локалния сървър. Той отговаря за получаването на заявките, изпратени от уеб приложението чрез мостовия API, и съответното им обработване. Това включва валидиране на заявката, проверка дали потребителят е удостоверен и упълномощен да извърши исканото действие и след това препращане на заявката към съответния компонент в локалния сървър.

След като заявката бъде обработена, BridgeAPIHandlingLogic връща отговор на уеб приложението чрез API на моста. Отговорът обикновено включва код на състоянието, указващ успеха или неуспеха на заявката, както и всички подходящи данни, които са били поискани.

### Диаграми на класовете на уеб приложението



14. Фигура 4.2.2. - Диаграма на класовете на уеб приложението

#### Описание по диаграмата

Предоставената диаграма на класове описва уеб приложение, което се състои от множество контролери и услуги, всеки от които отговаря за обработката на специфични функции. Приложението е създадено с помощта на архитектурния модел Model-View-Controller (MVC) и използва .NET Core.

Диаграмата включва шест класа, три интерфейса и три зависимости. Класовете включват AuthenticationController, BaseController, Controller, DevicesController, HomeController и DeviceData. Междувременно интерфейсите се състоят от IAuthenticationService, ICommunicationService и IDevicesService.

Класът AuthenticationController отговаря за обработката на удостоверяването на потребителя в рамките на уеб приложението. Този клас има две зависимости, IDataProtector и IAuthenticationService, които се използват съответно за защита на потребителските данни и за прилагане на функционалност, свързана с удостоверяване. Контролерът предоставя няколко метода, включително LogIn(), LogInLocalServer(), Register() и SignOut(), които позволяват на потребителите да влизат или да се регистрират в уеб приложението и да управляват състоянието на своята сесия.

Класът BaseController е основен контролер, който прилага необходимата логика, която всички други контролери трябва да използват. Този клас има една зависимост, SessionOptions, която се използва за управление на състоянието на потребителската сесия. Контролерът също така имплементира метода OnActionExecutionAsync(), който се изпълнява преди да бъде извикан метод за действие.

Класът DevicesController е отговорен за управлението на свързаната с устройството функционалност в рамките на уеб приложението. Този клас има две зависимости, IDataProtector и IDevicesService, които се използват съответно за защита на потребителските данни и изпълнение на функционалност за управление на устройства. Контролерът предоставя няколко метода, включително Index(), NoDevices(), SetChartArguments() и PostDataToDevice(), които позволяват на потребителите да управляват своите устройства и да визуализират данни.

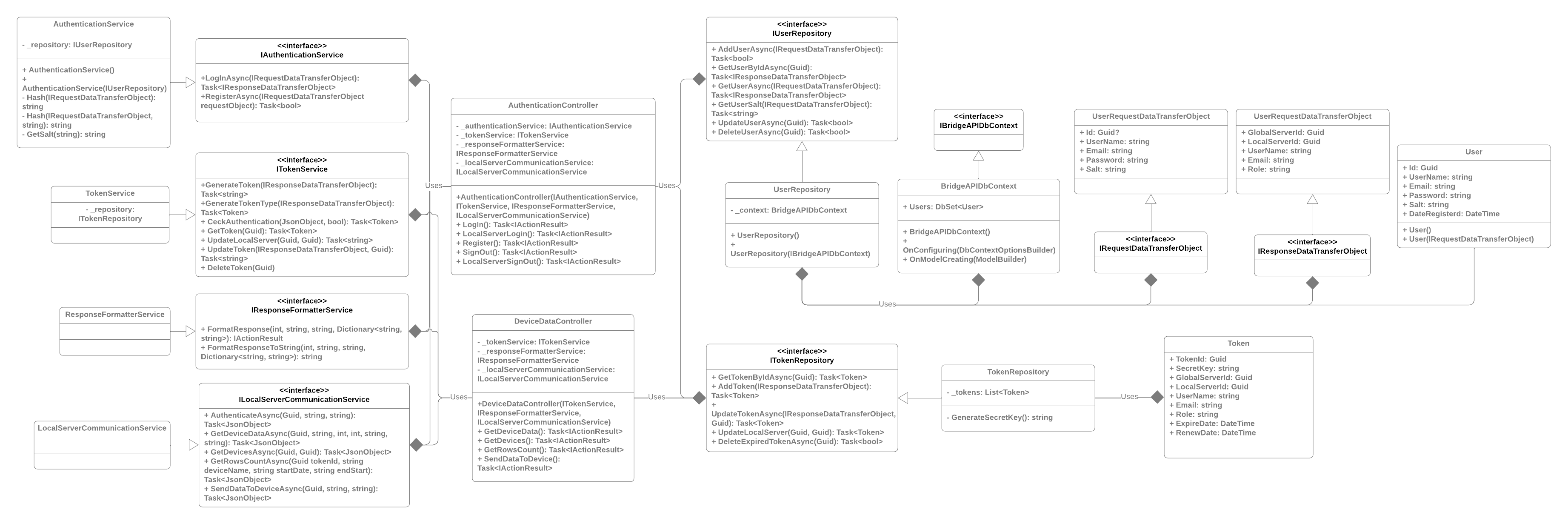
Класът HomeController отговаря за изобразяването на главните страници на уеб приложението. Този клас има една зависимост, ILogger<HomeController>, която се използва за регистриране на грешки и съобщения. Контролерът предоставя методи за изобразяване на страниците с индекс, поверителност и грешки.

Класът DeviceData представлява данните за конкретно устройство в рамките на уеб приложението. Този клас има три свойства, включително DeviceNames, Name и Infrastructure, които описват името на устройството и инфраструктурата, към която то принадлежи.

Освен това диаграмата на класовете включва три интерфейса: IAuthenticationService, ICommunicationService и IDevicesService. Тези интерфейси дефинират методите, които всеки съответен клас услуги трябва да реализира. Например IAuthenticationService дефинира методите LogInAsync(), RegisterAsync(), LogInLocalServerAsync(), SignOut() и LocalServerSignOut(), които се използват за удостоверяване на потребителите в рамките на уеб приложението.

В заключение, тази класова диаграма представлява уеб приложение, което е изградено с помощта на MVC архитектурния модел и .NET Core. Приложението се състои от множество контролери и услуги, всеки от които отговаря за обработката на специфични функции в рамките на уеб приложението. Контролерите зависят от конкретни услуги, а услугите реализират необходимите интерфейси. Като цяло тази диаграма предоставя ясно представяне на структурата на уеб приложението и връзките между неговите компоненти.

### Диаграми на класовете на BridgeAPI



214. Фигура 4.2.3. - Диаграма на класовете на уеб приложението

#### Описание по диаграмата

Класът AuthenticationController съдържа методи за управление на процесите на удостоверяване на потребителя, регистрация и излизане. Класът зависи от четири интерфейса: IAuthenticationService, ITokenService, IResponseFormatterService и ILocalServerCommunicationService. Тези интерфейси осигуряват функционалност за проверка на потребителските идентификационни данни, генериране и управление на токени, форматиране на съобщения за отговор и комуникация с локален сървър.

Класът DeviceDataController има методи за извличане и изпращане на данни към устройство, получаване на списък с устройства и получаване на общия брой редове в таблица с данни. Класът зависи от интерфейсите ITokenService, IResponseFormatterService и ILocalServerCommunicationService за обработка на базирано на токен удостоверяване, форматиране на отговори и комуникация с локалния сървър.

Интерфейсът IAuthenticationService предоставя два метода за удостоверяване на потребителски вход и заявки за регистрация. Тези методи са LogInAsync и RegisterAsync. LogInAsync проверява потребителските идентификационни данни и връща обект на отговор, съдържащ токен, ако влизането е успешно. RegisterAsync създава нов потребителски акаунт въз основа на предоставените идентификационни данни.

Интерфейсът ILocalServerCommunicationService предоставя методи за удостоверяване на потребител, получаване на данни за устройството, получаване на списък с устройства, получаване на общия брой редове в таблица с данни и изпращане на данни към устройство. Тези методи се използват от класовете AuthenticationController и DeviceDataController за комуникация с локален сървър.

Интерфейсът IResponseFormatterService предоставя два метода за форматиране на съобщения за отговор. Методът FormatResponse форматира съобщение за отговор като обект IActionResult, докато методът FormatResponseToString форматира съобщение за отговор като низ.

Интерфейсът ITokenService предоставя методи за генериране и управление на токени, включително създаване на нови токени, актуализиране на информация за токени и изтриване на изтекли токени. Интерфейсът също така предоставя методи за проверка на автентичността на токена и актуализиране на свързания с токена локален сървър.

Класът AuthenticationService имплементира интерфейса IAuthenticationService и предоставя методи за хеширане на пароли, извличане на солни стойности и създаване на потребителски акаунти. Класът зависи от интерфейса IUserRepository за управление на потребителски данни.

Класът LocalServerCommunicationService имплементира интерфейса ILocalServerCommunicationService и предоставя методи за комуникация с локален сървър. Тези методи включват удостоверяване на потребител, получаване на данни за устройството, получаване на списък с устройства, получаване на общия брой редове в таблица с данни и изпращане на данни до устройство.

Класът ResponseFormatterService имплементира интерфейса IResponseFormatterService и предоставя методи за форматиране на съобщения за отговор.

Класът TokenService имплементира интерфейса ITokenService и предоставя методи за генериране и управление на токени. Класът зависи от интерфейса на ITokenRepository за управление на данни за токени.

Класът BridgeAPIDbContext представлява контекст на база данни и осигурява достъп до таблицата User. Класът отговаря за създаването и конфигурирането на схемата на базата данни.

Класът Token представлява обект на токен и съдържа информация за идентификатора на токена, секретния ключ, свързаните сървъри, потребителска информация и дати на изтичане.

Класът Потребител представлява потребителски обект и съдържа информация за ИД на потребителя, потребителско име, имейл адрес, парола, солна стойност и дата на регистрация.

Интерфейсът ITokenRepository предоставя методи за добавяне, актуализиране и изтриване на токени, както и извличане на токени по ID.

Интерфейсът IUserRepository предоставя методи за добавяне, актуализиране и изтриване на потребители, както и извличане на потребителски данни.

### Описание на слоевете и тяхната функция

Системата за домашна сигурност, модернизация и наблюдение на здравето на Интернет на нещата е проектирана да предостави на потребителите адаптивна и разширяема платформа за наблюдение и контрол на техните устройства. Системата е изградена с помощта на трислойна архитектура, която включва локален сървър, BridgeAPI, уеб приложение и най-важните, от които интернет на нещата устройства.

Интернет на нещата устройства: Това са физическите устройства, които са част от системата за домашна сигурност, модернизация и мониторинг на здравето на Интернет на нещата. Тези устройства могат да бъдат сензори, камери или друг тип устройство, което е в състояние да генерира данни за домашната среда или здравето на хората в нея. Устройствата се програмират с помощта на Arduino IDE с C++ и комуникират с локалния сървър чрез TCP връзка.

Локалният сървър е GUI приложение, което позволява на потребителите да конфигурират ролите и разрешенията на всеки потребител в системата. Това е важно от съображения за сигурност, тъй като потребителите може да имат различни нива на достъп до определени устройства. Например, дете в домакинството не трябва да има достъп до камерата в спалнята на родителя. Локалният сървър позволява на потребителите лесно да конфигурират и управляват тези разрешения, осигурявайки удобен за потребителя интерфейс за този ключов аспект на системата.

BridgeAPI е отговорен за получаване на данните от локалния сървър и предоставянето им на уеб приложението. Този API е изграден с помощта на ASP.NET Core MVC, което позволява стандартизирани заявки и отговори. Това означава, че потребителите могат да създават свои собствени приложения, които могат да комуникират със системата, като използват предварително зададения брой операции и стандартните отговори на API. Това прави системата силно адаптивна, тъй като потребителите могат да създават свои собствени приложения за наблюдение и контрол на своите устройства, вместо да разчитат единствено на уеб приложението, предоставено със системата.

Уеб приложението е мястото, където потребителите могат да преглеждат данните на устройството и да правят заявки към BridgeAPI. Уеб приложението също е изградено с помощта на ASP.NET Core MVC, което улеснява комуникацията с API. Уеб приложението предоставя лесен за използване интерфейс за наблюдение на данните на устройството и контрол на устройствата, което го прави важен аспект от системата за повечето потребители.

Комуникацията между слоевете е защитена и както уеб приложението, така и локалният сървър трябва да се удостоверят в BridgeAPI, за да функционират. Това осигурява силни мерки за сигурност за защита на потребителските данни и предотвратяване на неоторизиран достъп до системата.

Системата е проектирана да бъде много разширяема, позволявайки на потребителите да добавят нови устройства, ако е необходимо. Когато се добави ново устройство, автоматично се създава нова таблица в базата данни на SQL Server, използвана за съхраняване на данните за устройството. Това улеснява потребителите да персонализират своите системи, за да отговорят на техните специфични нужди.

В обобщение, Интернет на нещата системата за домашна сигурност, модернизация и мониторинг на здравето се състои от физически Интернет на нещата устройства, които генерират данни за домашната среда и здравето на нейните обитатели. Локалният сървър действа като централен център на системата, като събира и съхранява данните, генерирани от Интернет на нещата устройствата в базата данни на SQL Server. BridgeAPI предоставя сигурен RESTful API за достъп до данните, съхранени в базата данни, а уеб приложението предоставя удобен за потребителите интерфейс за достъп и наблюдение на техните устройства и здравни данни. Заедно тези компоненти образуват мощна система, която може да помогне на потребителите да наблюдават домовете и здравето си в реално време и да предприемат действия, ако е необходимо.

### Комуникация между локания сървър и интернет на нещата устройствата

Комуникацията между устройствата и локалния сървър се осъществява чрез TCP протокол. Когато ново устройство се свърже с локалния сървър, то се прекъсва и се добавя към таблицата с устройства в базата данни. Това се прави, за да се следят всички устройства, които са свързани към системата.

След като устройството бъде добавено към базата данни, потребителят трябва да одобри устройството, за да може то да се свърже със сървъра. Това е мярка за сигурност, въведена за предотвратяване на неоторизирани устройства от свързване към системата.

След като потребителят одобри устройството, устройството може да се свърже с локалния сървър. Първото съобщение, което устройството трябва да изпрати, е низ за удостоверяване. Низът за удостоверяване трябва да съдържа името на устройството и колоните в таблицата, от които се нуждае, за да съхранява своите данни в базата данни.

След това локалният сървър ще създаде желаната таблица за устройството въз основа на информацията, предоставена в низа за удостоверяване. Този процес гарантира, че данните на устройството се съхраняват в правилната таблица и са лесно достъпни, когато е необходимо.

След процеса на удостоверяване устройството може да започне да изпраща данни към локалния сървър. Данните се изпращат под формата на JSON пакети по TCP протокола. Локалният сървър получава пакетите и анализира JSON данните, за да извлече съответната информация.

След като данните бъдат извлечени, локалният сървър ги съхранява в съответната таблица в базата данни. След това потребителят може да получи достъп до тези данни чрез уеб приложението, което прави заявки към BridgeAPI на локалния сървър. BridgeAPI получава заявката и извлича данните от базата данни. След това данните се връщат към уеб приложението под формата на JSON отговор.

Като цяло този процес на комуникация гарантира, че данните, изпратени от устройствата, се съхраняват сигурно в базата данни и могат да бъдат лесно достъпни от потребителя чрез уеб приложението. Използването на TCP протокола гарантира, че данните се предават надеждно и сигурно между устройствата и локалния сървър.

### Комуникация между BridgeAPI и уеб приложението

Комуникацията между уеб приложението и BridgeAPI е важна част от цялата система. Той позволява на потребителя да взаимодейства със своите интернет на нещата устройства по безпроблемен и интуитивен начин. Позволете ми да ви го разкажа подробно.

Уеб приложението и BridgeAPI комуникират чрез RESTful апликационен програмистки интерфейс заявки. Това означава, че уеб приложението изпраща HTTP заявки към BridgeAPI, който след това отговаря с исканите данни. BridgeAPI действа като посредник между уеб приложението и интернет на нещата устройствата, обработвайки всички заявки и отговори.

Когато потребителят отвори уеб приложението, то изпраща заявка до BridgeAPI за списък с налични устройства. BridgeAPI извлича списъка с устройства от базата данни и го връща в уеб приложението като JSON обект. След това уеб приложението показва списъка с устройства на потребителя.

Когато потребителят избере устройство, уеб приложението изпраща заявка до BridgeAPI за данните на устройството. BridgeAPI извлича данните за устройството от базата данни и ги връща в уеб приложението като JSON обект. След това уеб приложението показва данните за устройството на потребителя.

Ако потребителят иска да извърши действие на устройство, като например да го включи или изключи, уеб приложението изпраща заявка до BridgeAPI с желаното действие и ID на устройството. След това BridgeAPI изпраща заявката за действие до IoT устройството, което изпълнява действието и изпраща отговор обратно на BridgeAPI. След това BridgeAPI връща отговора на уеб приложението като JSON обект.

Комуникацията между уеб приложението и BridgeAPI е защитена чрез HTTPS протокол. Това гарантира, че всички данни, прехвърляни между двете, са криптирани и защитени. Освен това BridgeAPI прилага удостоверяване, базирано на токени, за да гарантира, че само оторизирани потребители имат достъп до системата.

BridgeAPI е проектиран да обработва множество заявки едновременно, което го прави ефективен и мащабируем. Той може да обработва заявки от множество потребители и устройства едновременно, без да компрометира производителността.

Като цяло комуникацията между уеб приложението и BridgeAPI е съществена част от системната архитектура. Той позволява на потребителя да взаимодейства със своите интернет на нещата устройства по удобен за потребителя начин, докато BridgeAPI обработва всички сложни операции във фонов режим.

### Комуникация между BridgeAPI и локалния сървър

Както бе споменато по-рано, локалният сървър и BridgeAPI комуникират чрез протокола за контрол на предаването (TCP). Това е надежден, ориентиран към свързване протокол, който осигурява последователен и стабилен комуникационен канал между двете единици.

Процесът на комуникация започва с локалния сървър, който установява TCP връзка с BridgeAPI, използвайки IP адреса и номера на порта на API. След като връзката бъде установена, двете организации могат да обменят данни.

Локалният сървър използва персонализиран протокол за изпращане на данни към BridgeAPI. Този протокол включва заглавие и полезен товар. Заглавието съдържа информация за типа и размера на полезния товар, както и друга подходяща информация, като например адресите на източника и местоназначението.

Полезният товар съдържа действителните данни, които локалният сървър иска да изпрати на BridgeAPI. Тези данни могат да бъдат във всякакъв формат, като двоичен или текстов, в зависимост от нуждите на приложението.

BridgeAPI, при получаване на данните от локалния сървър, проверява заглавката, за да определи типа и размера на полезния товар. След това извлича полезния товар и го обработва съответно.

По същия начин, когато BridgeAPI трябва да изпрати данни към локалния сървър, той използва същия персонализиран протокол. Той създава заглавка, която включва информация за полезния товар, и след това прикачва самия полезен товар. Полезният товар може да бъде всякакви данни, като актуализации на състоянието, показания на сензори или друга подходяща информация.

Локалният сървър, при получаване на данните от BridgeAPI, проверява заглавката, за да определи типа и размера на полезния товар. След това извлича полезния товар и го обработва съответно.

Този процес на обмен на данни между локалния сървър и BridgeAPI се извършва непрекъснато, когато нови данни станат достъпни или когато се изискват актуализации. Това гарантира, че локалният сървър винаги разполага с най-актуалната информация за интернет на нещата устройствата и техния статус и може да реагира съответно.

Използването на TCP като основен протокол гарантира, че комуникацията между локалния сървър и BridgeAPI е надеждна и последователна. TCP предоставя функции като проверка на грешки, контрол на потока и контрол на задръстванията, които гарантират, че данните се предават точно и ефективно.

Освен това персонализираният протокол, използван от локалния сървър и BridgeAPI, позволява обмен на всякакъв тип данни във всякакъв формат, което го прави гъвкаво и адаптивно решение за всяко интернет на нещата приложение.

В обобщение, комуникацията между локалния сървър и BridgeAPI с помощта на TCP е надеждно, последователно и гъвкаво решение за обмен на данни между двете единици. Използването на персонализиран протокол позволява обмен на всякакъв тип данни, докато TCP гарантира, че комуникацията е точна и ефективна.

## Организация и код на заявките към база от данни

Когато локалният сървър изпрати заявка към базата данни, тя преминава през персонализираната обектно релационен картограф (ORM) система, разработена за този конкретен проект. Обектно релационната картографска (ORM) системата е отговорна за обработката на всички взаимодействия с базата данни, включително създаване на таблици за нови устройства, съхраняване на данни от устройства и извличане на данни за уеб приложението за показване на потребителя.

Системата обектно релационен картограф (ORM) работи, като използва низа за удостоверяване, предоставен от Интернет на нещата устройството, за да определи името на таблицата и колоните, необходими за съхраняване на данните. Тази информация се използва за динамично създаване на SQL заявки, които могат да бъдат изпратени до базата данни.

Когато локалният сървър изпрати заявка до обектно релационен картограф (ORM) системата, той указва какво действие иска да извърши (напр. извличане на данни от конкретна таблица, вмъкване на нови данни в таблица и т.н.). След това обектно релационен картограф (ORM) системата генерира подходящата SQL заявка въз основа на исканото действие и я изпраща към базата данни.

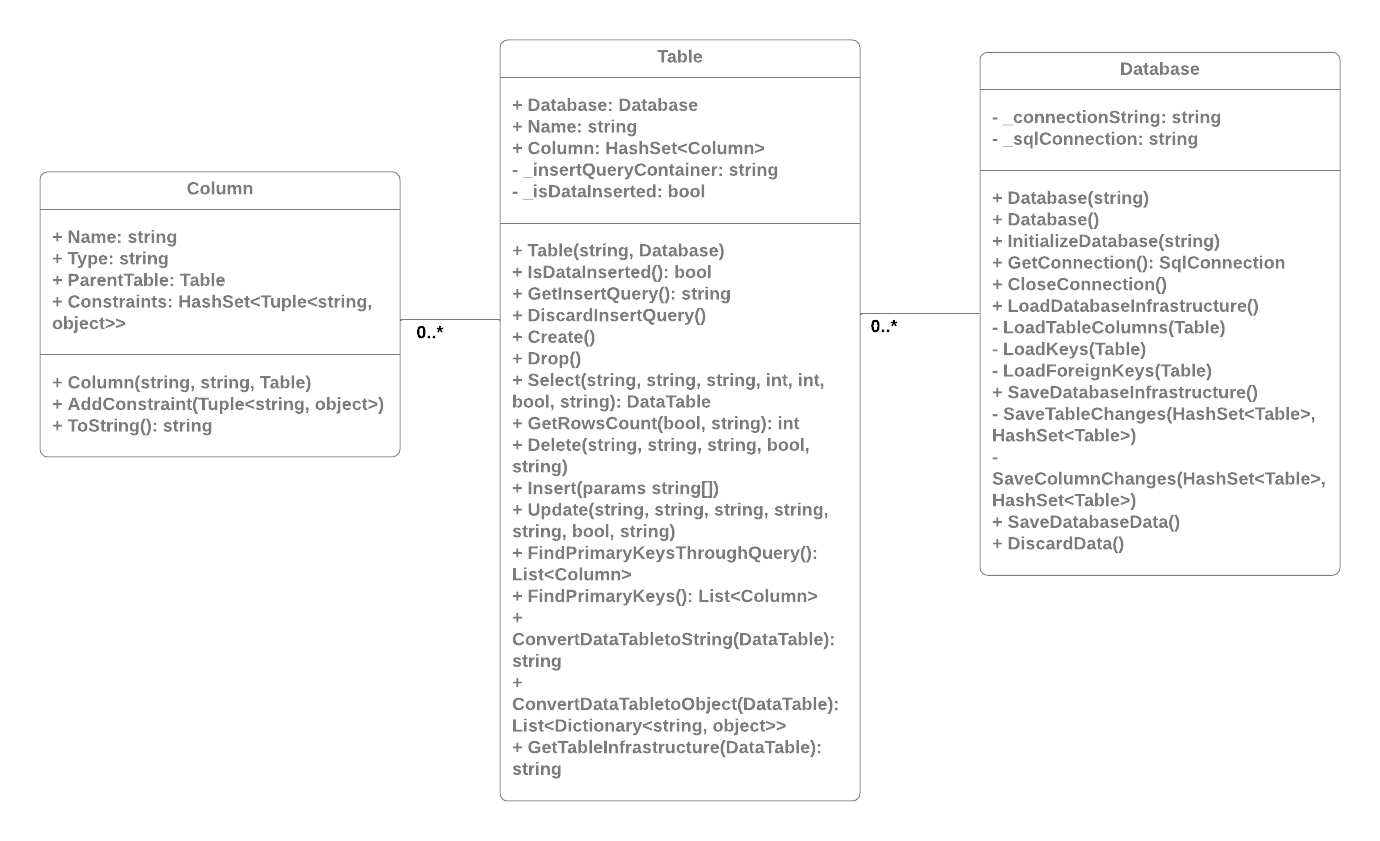
След като базата данни получи заявката, тя я обработва и изпраща исканите данни обратно към обектно релационен картограф (ORM) системата. След това обектно релационен картограф (ORM) системата обработва данните, преобразувайки ги от необработен SQL резултат във формат, който може лесно да се използва от локалния сървър. След това обработените данни се връщат на локалния сървър, който може да ги използва за актуализиране на уеб приложението или извършване на други действия, ако е необходимо.

Един важен аспект на обектно релационен картограф (ORM) системата е нейната способност да обработва грешки, които могат да възникнат по време на процеса на взаимодействие с базата данни. Ако възникне грешка, обектно релационен картограф (ORM) системата ще я улови и ще предостави ясно съобщение за грешка на локалния сървър, който след това може да покаже съобщението за грешка на потребителя или да предприеме друго подходящо действие.

Като цяло персонализираната обектно релационен картограф (ORM) система играе критична роля във функционирането на системата за домашна сигурност, модернизация и мониторинг на здравето на Интернет на нещата, тъй като осигурява рационализиран и ефективен начин за взаимодействие с базата данни, като гарантира, че данните се съхраняват и извличат точно и надеждно .

### Персонализираната обектно релационен картограф (ORM) система

#### Диаграма на класовете на персонализираната обектно релационен картограф система(ORM)



14. Фигура 4.3.1.1. - Диаграма на класовете на персонализирания обектно релационен картограф

##### Описание по диаграмата

Персонализираният обектно релационен картограф (ORM) е рамка, използвана за опростяване на процеса на достъп и управление на база данни. Персонализираният ORM, който споменахте, се състои от три основни класа: база данни, таблица и колона.

Класът Database е отговорен за свързването към базата данни и управлението на връзката. Той съдържа методи за изпълнение на заявки и извличане на данни от базата данни. Класът Database също обработва картографирането на обекти към таблици и колони на база данни.

Класът Table представлява таблица от база данни. Той съдържа методи за извършване на CRUD (създаване, четене, актуализиране, изтриване) операции върху данните в таблицата. Класът Table също обработва картографирането на обекти към колоните в таблицата.

Класът Column представлява колона в таблица на база данни. Той съдържа информация за колоната, като нейното име, тип данни и ограничения. Класът Column също управлява картографирането на свойствата на обекта към колоната в таблицата.

#### Работния процес на персонализираната системата обектно релационен картограф (ORM)

Персонализираната обектно релационен картограф (ORM) система, използвана в локалния сървър, е критичен компонент на гъвкавостта и разширяемостта на системата. Целта му е да предостави начин на потребителя да добавя нови устройства и да конфигурира как данните от тези устройства се съхраняват в базата данни. Системата обектно релационен картограф (ORM) е проектирана да улесни разработчиците да добавят нови таблици към базата данни, без да се налага да пишат сложни SQL изрази или да взаимодействат директно с базата данни. Вместо това системата разчита на низа за удостоверяване, предоставен от всяко Интернет на нещата устройство, за да определи как трябва да се съхраняват данните.

Низът за удостоверяване съдържа името на таблицата и колоните, необходими за съхраняване на данните на устройството. Обектно релационен картограф (ORM) системата използва тази информация, за да създаде нова таблица в базата данни с указаното име и колони. По подразбиране системата дефинира три колони за всяка таблица: уникален идентификатор (Id, int), създаден (Created, datetime(2)) и уникалния идентификатор на устройството (DeviceId, int) като външен ключ, препращащ към таблицата на устройството. Тези колони по подразбиране осигуряват последователност в данните, съхранявани в базата данни, и помагат за предотвратяване на загуба на данни.

За да създаде новата таблица, персонализираната обектно релационен картограф (ORM) система използва динамичен SQL, за да изпълни оператора CREATE TABLE. Този подход гарантира, че системата е достатъчно гъвкава, за да се справи с всяка конфигурация на таблица, без да се налага да кодирате предварително фиксиран набор от таблици.

### Обектно релационен картограф (ORM) – Entity Framework Core

От друга страна, BridgeAPI използва EF Core, популярна обектно релационен картограф (ORM) рамка с отворен код за .NET. EF Core предоставя удобен начин за взаимодействие с базата данни, като позволява на разработчиците да работят с обекти и техните връзки, вместо директно да взаимодействат с основните таблици на базата данни. Това улеснява прилагането на сложни заявки и гарантира сигурността на данните чрез предотвратяване на атаки чрез SQL инжектиране.

EF Core използва подход на първо място с кода за проектиране на база данни, което означава, че схемата на базата данни на системата е дефинирана в C# код, а не в SQL скриптове. Това улеснява извършването на промени в схемата на базата данни и гарантира, че схемата винаги е в синхрон с кода. Освен това EF Core предоставя редица други предимства, като автоматично управление на транзакции и вградена поддръжка за каширане и контрол на паралелността.

Чрез използването на персонализирана обектно релационен картограф (ORM) система за локалния сървър и EF Core за BridgeAPI, системата гарантира, че може да се справи както с гъвкавите нужди за съхранение на Интернет на нещата устройства, така и с изискванията за сигурност на едно модерно уеб приложение. Този подход позволява на системата да бъде гъвкава и адаптивна към нови устройства и променящи се изисквания, като същевременно гарантира сигурността и целостта на данните.

## Наличие на потребителски интерфейс (конзолен, графичен, уеб)

Започвайки с локалния сървър на GUI, потребителският интерфейс има табло за управление, показващо подходяща информация като броя на потребителите, свързаните устройства и скорошни събития. Интерфейсът трябва също така да позволява на потребителя да добавя, променя или изтрива потребителски акаунти, включително техните роли и разрешения.

За локалния CLI сървър интерфейсът може да се състои от интерфейс на командния ред с опции за добавяне, модифициране и изтриване на потребителски акаунти. Интерфейсът на командния ред e проектиран с ясни и кратки инструкции, за да помогне на потребителя да навигира в системата.

В уеб приложението потребителският интерфейс ще се съсредоточи основно върху наблюдението на данните на устройството, с табло за управление, показващо информация в реално време на данните на устройствата. Това включва графики, диаграми и други визуални представяния на данни. Интерфейсът може също така да позволи на потребителя да задава предупреждения и известия за конкретни събития или промени в данните на устройството.

# Ефективност и бързодействие на решението

По отношение на изпълнението на код, този вид приложение би било много ефективно, защото използва съвременни технологии като .NET Core, TCP протокол и персонализиран ORM. .NET Core е известен със своята производителност и мащабируемост, което го прави чудесен избор за изграждане на високопроизводителни приложения. TCP протоколът е широко използван заради своята надеждност, а персонализираният ORM гарантира ефективно съхранение и извличане на данни.

От гледна точка на архитектурния дизайн, това приложение е проектирано с мисъл за мащабируемост и разширяемост, което позволява на потребителите да добавят толкова устройства, колкото желаят, и да конфигурират как данните за устройството да се записват в базата данни. Използването на BridgeAPI също позволява лесна интеграция с други приложения и услуги, което го прави универсално решение за мониторинг на Интернет на нещата данни.

В сравнение с други подобни решения, това приложение се отличава със своята персонализирана ORM система, която позволява лесно добавяне на нови устройства и полета с данни без необходимост от сложни миграции на бази данни. Използването на TCP протокол за комуникация на устройството гарантира надежден трансфер на данни, докато уеб приложението предоставя интуитивен потребителски интерфейс за наблюдение на данните на устройството в реално време.

Други подобни решения могат да използват различни технологии и протоколи, като MQTT или WebSocket за комуникация на устройства и SQL или NoSQL бази данни за съхранение на данни. Въпреки това персонализираната ORM система, използвана в това приложение, предоставя уникално предимство пред други решения, като позволява лесно управление и разширяване на схемата на базата данни. Освен това, BridgeAPI позволява интеграция с широк набор от други приложения и услуги, като допълнително подобрява гъвкавостта на приложението.

# Тестване

В сложна система като описаната има няколко компонента, които трябва да бъдат тествани, за да се гарантира правилното функциониране на приложението като цяло. Някои от ключовите области, които трябва да бъдат тествани, са:

## Видове тестване

Функционално тестване: Този тип тестване гарантира, че системата отговаря на функционалните изисквания и спецификации. Това включва проверка дали всяка характеристика и функция на системата работи правилно и изпълнява предназначената си задача, както е описано в документа с изискванията. Това включва тестване на различните функции на уеб приложението като влизане, регистрация, управление на потребителите и визуализация на данни, както и функциите на интернет на нещата устройствата като събиране, предаване и съхранение на данни от сензори.

Тестване на интеграцията: Този тип тестване проверява дали различните компоненти на системата са интегрирани правилно и комуникират помежду си според очакванията. Интегрираното тестване помага да се идентифицират всички проблеми, които могат да възникнат при комбиниране на различни компоненти и гарантира, че системата работи като единна, сплотена единица. Това включва тестване на комуникацията между уеб приложението и локалния сървър, както и между локалния сървър и интернет на нещата устройствата.

Тестване на производителността: Този тип тестване оценява способността на системата да обработва голямо количество трафик и данни, като същевременно поддържа приемливи нива на производителност. Тестването на производителността помага да се идентифицират тесните места и други проблеми, които могат да възникнат, когато системата е под голямо натоварване. Това включва тестване на времето за реакция на уеб приложението, скоростта, с която локалният сървър може да извлича и обработва данни, и ефективността на интернет на нещата устройствата при събиране и предаване на данни.

Тестване на сигурността: Този тип тестване гарантира, че системата е сигурна и защитена срещу потенциални заплахи като кибер атаки, пробиви на данни и други форми на кибер атаки. Тестването на сигурността включва идентифициране на уязвимости в системата и прилагане на мерки за предотвратяване на тяхното използване. Това включва тестване на процесите за удостоверяване и оторизация на уеб приложението, както и процесите на криптиране и декриптиране, използвани за осигуряване на предаване на данни между компонентите на системата.

Тестване на използваемостта: Този тип тестване оценява лекотата на използване на системата и потребителското изживяване. Тестването за използваемост помага да се идентифицират всички проблеми, които могат да направят системата трудна или объркваща за навигацията на потребителите, както и възможности за подобряване на цялостното потребителско изживяване. Това включва тестване на потребителския интерфейс на уеб приложението, както и функционалността и лекотата на използване на интернет на нещата устройствата.

В обобщение, тестването на системата трябва да обхваща тестове за функционалност, интеграция, производителност, сигурност и използваемост, за да се гарантира, че системата работи правилно, е защитена и осигурява добро потребителско изживяване. Чрез щателно тестване на тези области системата може да бъде оптимизирана за максимална производителност и надеждност, като същевременно минимизира риска от потенциални проблеми или уязвимости.

## План за тестване

За функционалното тестване на приложението е използвана работната рамка NUnit. NUnit е популярна рамка за модулно тестване с отворен код за .NET езици като C#. Той предоставя прост и лесен за използване синтаксис за писане и изпълнение на автоматизирани тестове и поддържа широка гама от типове тестове, включително тестове за единици, интеграционни тестове и тестове за приемане. NUnit позволява на разработчиците да пишат тестови случаи, които могат да се изпълняват автоматично, което улеснява откриването на грешки в началото на процеса на разработка и гарантира, че промените в кода не въвеждат регресии. Рамката също така предоставя различни твърдения, тестове и инструменти за отчитане, за да помогне на разработчиците бързо да идентифицират и диагностицират проблеми в техния код.

Главните функционалности, които са тествани са персонализирания обектно релационен картограф, неговите класове и методи, дата манипулативната логика, която отговаря за създаване, четене, обновяване и изтриване на данни от базата от данни, логиката за удостоверяване и упълномощаване на потребителя. Всяка функционалност е покрита с осемдесет или повече процента от тестове, като те биват наименувани по следния начин: Трябва\_ДействиетоКоетоТрябваДаСеСлучи\_Когато\_МетодаКойтоБиваИзвикван

При тестване на функционалност се преглеждат три главни аспекта: дали функционалността, която се тества, работи както е предвидено, дали функционалността приема грешни входни данни и как реагира на тях, дали функционалността връща грешки при определени условия. Тествайки един метод с тези три аспекта гарантираме, че функционалността бива покрита с осемдесет или повече процента тестове.

Един тест бива разделен на три основни части: нареждане, действие, твърдение. В секцията на нареждането се зареждат данните и потенциалните връзки, който функционалността изисква. В секцията на действието изпълнява функционалността и се запазва резултата от изпълнението на метода и накрая в твърдението проверяваме дали резултата отговаря на очаквания резултат

Пример: Тестване на функционалността за регистрация на потребител в локалния сървър.

При тестването на тази функционалност трябва да се направят два теста. Първия, който да тества самата функционалност и втория, който се фокусира върху, какво ще бъде поведението на метода или функционалността, ако входните данни не са форматирани както трябва.

# Заключение и възможно бъдещо развитие

## Заключение

В този проект проучихме дизайна и внедряването на модулна Интернет на нещата система за домашна сигурност, която включва както функции за сигурност, така и функции за наблюдение на здравето. Чрез този проект ние демонстрирахме потенциала на Интернет на нещата технологията за подобряване на безопасността и благосъстоянието на собствениците на жилища и техните семейства чрез създаване на адаптивна и свързана система от сензори и устройства.

Чрез проектирането на удобно за потребителя приложение, собствениците на жилища могат лесно да настроят и персонализират своята модулна Интернет на нещата система за домашна сигурност, за да отговарят на техните уникални нужди и предпочитания. Софтуерът и свързаността, необходими, за да се гарантира, че всички устройства комуникират ефективно, са разгледани подробно по време на проекта, което позволява на собствениците на жилища да създадат наистина модерен и свързан дом, където всички устройства и сензори работят безпроблемно заедно, за да осигурят повишена безопасност и благополучие.

Потенциалните потребители на модулно Интернет на нещата приложение за домашна сигурност и мониторинг на здравето включват собственици на жилища, членове на семейството, лица, които се грижат за тях, и специалисти по домашна сигурност. Приложението е проектирано да бъде достъпно и удобно за всеки, който иска да подобри безопасността и благосъстоянието на дома и близките си.

В бъдеще ще са необходими по-нататъшни изследвания и разработки в областта на Интернет на нещата домашната сигурност и мониторинг на здравето, за да продължим да подобряваме функционалността и използваемостта на тези системи. Този проект обаче демонстрира потенциала на Интернет на нещата технологията да създаде наистина модерен и свързан дом, повишаващ безопасността, сигурността и благосъстоянието за собствениците на жилища и техните семейства. Чрез включването на функции за наблюдение на здравето в системата, собствениците на жилища могат не само да подобрят личното си здраве, но и да имат спокойствие, знаейки, че домът им е защитен и личното им здраве се наблюдава и проследява в реално време.

## Бъдещо развитие

Бъдещите планове и реализация на модулен Интернет на нещата проект за домашна сигурност и мониторинг на здравето може да включва следното:

1. Разширяване на възможностите на системата: Тъй като технологията продължава да се развива, бъдещите планове за модулен Интернет на нещата проект за домашна сигурност и мониторинг на здравето може да включват разширяване на възможностите на системата, за да включва допълнителни сензори и устройства. Например, системата може да интегрира по-усъвършенствани функции за наблюдение на здравето или да включва нови функции за сигурност, които използват нововъзникващи технологии като изкуствен интелект или машинно обучение.
2. Подобряване на потребителското изживяване: Текущите подобрения на потребителското изживяване ще бъдат ключово съображение за проекта. Бъдещите планове може да включват усъвършенстване на потребителския интерфейс, рационализиране на процеса на настройка и подобряване на цялостната използваемост на системата. Тези подобрения ще помогнат да се гарантира, че потребителите могат лесно и ефективно да наблюдават своите домове и лични здравни показатели в реално време.
3. Интегриране с други интелигентни домашни устройства: Модулният характер на Интернет на нещата системата за домашна сигурност означава, че тя може лесно да се интегрира с други интелигентни домашни устройства. Бъдещите планове за проекта може да включват разширяване на гамата от устройства, с които системата може да се интегрира, като интелигентни ключалки, гласови асистенти и системи за забавление. Това ще осигури на потребителите безпроблемно и свързано изживяване в целия им дом.
4. Мащабиране на системата: Тъй като проектът набира сила и става по-популярен, мащабирането на системата, за да поддържа по-голяма потребителска база, ще бъде важно съображение. Бъдещите планове може да включват разширяване на инфраструктурата, използвана за поддръжка на системата, като например инвестиране в допълнителни сървъри или базирани на облак ресурси за справяне с увеличен трафик. Това ще гарантира, че системата остава надеждна и ефективна, докато расте.

Като цяло, бъдещите планове и реализация за модулен Интернет на нещата проект за домашна сигурност и мониторинг на здравето вероятно ще включват непрекъснати подобрения на възможностите на системата, потребителското изживяване и мерките за сигурност. Като е в крак с най-новите Интернет на нещата технологии и непрекъснато обновява, проектът има потенциала да се превърне във водещо решение за домашна сигурност и наблюдение на личното здраве.

# Използвани литературни източници и Уеб сайтове

Microsoft. ASP.NET Documentation, [онлайн] Достъпно на: [https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0](https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0%20) [Последно достъпен на 10.04.2023]

Microsoft. .NET Documentation, [онлайн] Достъпно на: <https://learn.microsoft.com/en-us/dotnet/> [Последно достъпен на 07.03.2023]

Microsoft. ASP.NET Core web API documentation with Swagger / OpenAPI, [онлайн] Достъпно на: [https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-7.0](https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-7.0%20) [Последно достъпен на 01.04.2023]

# Приложения

Тази документация е разгледана теоретичната част на системата за домашна сигурност и мониторинг на здравето. За по подобна информация от гледна точка на функционалност и имплементация е направена отделна HTML документация, в която биват описани всички слоеве, техните методи, какво правят те и как те взаимодействат помежду си. Нея може да откриете в директорията “Documentation/html”, която се намира в главната папка на проекта.

# Критерии и показатели за оценяване

|  |  |  |
| --- | --- | --- |
| Критерии и показатели за оценяване | Максимален брой точки за показателите | Максимален брой точки за критерия |
| 1. Съответствие с изискванията за съдържание и структура на дипломния проект |  | 20 |
| 1. 1. логическа последователност и структура на изложението, балансиране на отделните части | 4 |  |
| 1.2. задълбоченост и пълнота при формулиране на обекта, предмета, целта и задачите в разработването на темата | 7 |  |
| 1.3. използване на подходящи изследователски методи | 4 |  |
| 1.4. стил и оформяне на дипломната работа (терминология, стил на писане, текстообработка и оформяне на фигури и таблици) | 5 |  |
| 2. Съответствие между поставените цели на дипломния проект и получените резултати |  | 20 |
| 2.1. изводите следват пряко от изложението, формулирани са ясно, решават поставените в началото на изследването цели и задачи и водят до убедителна защита на поставената теза | 10 |  |
| 2.2. оригиналност, значимост и актуалност на темата | 6 |  |
| 2.3. задълбоченост и обоснованост на предложенията и насоките | 4 |  |
| 3. Представяне на дипломния проект |  | 20 |
| 3.1. представянето на разработката по темата e ясно и точно | 5 |  |
| 3.2. онагледяване на експозето с:  а) презентация;  б) графични материали;  в) практически резултати;  г) компютърна мултимедийна симулация и анимация | 10 |  |
| 3.3. умения за презентиране | 5 |  |
| 4. Отговори на зададените въпроси от рецензента и/или членовете на комисията за защита на дипломен проект |  | 30 |
| 4.1. разбира същността на зададените въпроси и отговаря пълно, точно и убедително | 10 |  |
| 4.2. логически построени и точни отговори на зададените въпроси | 10 |  |
| 4.3. съдържателни и обосновани отговори на въпросите | 10 |  |
| 5. Използване на професионалната терминология, добър и ясен стил, обща езикова грамотност |  | 10 |
| 5.1. Правилно използване на професионалната терминология | 5 |  |
| 5.2. Ясен изказ и обща езикова грамотност | 5 |  |
| Общ брой точки: | Максимален бр. точки 100 | Максимален бр. точки 100 |