**ДИПЛОМЕН ПРОЕКТ**

**НА ТЕМА**

**ПЛАТФОРМА ЗА ДОБРОВОЛНА ВЗАИМНОПОМОЩ ЗА ДОМАШНИ ЛЮБИМЦИ**

**“Pet Exchange”**

**Ученик: Ръководител консултант:**

Виктор Божидаров Зашев д-р. Янислав Картелов

Професия: Системен програмист

Специалност: Системно програмиране

гр. Пловдив, 2025

Съдържание

[I. Увод 4](#_Toc193888241)

[II. Основна част 5](#_Toc193888242)

[1. Цел на ДП. 5](#_Toc193888243)

[2. Проучване на съществуващи приложения. 7](#_Toc193888244)

[3. Обхват на проекта 8](#_Toc193888245)

[4. Технологии 10](#_Toc193888246)

[4.1 .NET 8 10](#_Toc193888247)

[4.2 Visual Studio 2022 11](#_Toc193888248)

[4.3 C# 11](#_Toc193888249)

[4.4 ASP.NET CORE 11](#_Toc193888250)

[4.5 MSSQL Server 12](#_Toc193888251)

[4.6 MSSQL Server Management Studio 12](#_Toc193888252)

[4.7 Entity Framework Core 13](#_Toc193888253)

[4.8 LINQ 13](#_Toc193888254)

[4.9 ASP.NET Core Identity 14](#_Toc193888255)

[4.10 Уеб браузър 14](#_Toc193888256)

[4.11 HTML 14](#_Toc193888257)

[4.12 CSS 15](#_Toc193888258)

[4.13 Javascript 16](#_Toc193888259)

[4.14 Razor 17](#_Toc193888260)

[4.15 Bootstrap Framework 17](#_Toc193888261)

[4.16 NUnit 18](#_Toc193888262)

[5. Реализация 19](#_Toc193888263)

[5.1 Функционалност 19](#_Toc193888264)

[5.1.1 База Данни 21](#_Toc193888265)

[UML CLASS DIAGRAM 21](#_Toc193888266)

[DATABASE SCREENSHOT 21](#_Toc193888267)

[5.2 Системни изисквания 22](#_Toc193888268)

[5.3 Експлоатация 22](#_Toc193888269)

[5.4 Архитектурен дизайн 22](#_Toc193888270)

[6. Потребителска документация 42](#_Toc193888271)

[6.1 Ръководство за потребителя 42](#_Toc193888272)

[6.2 Ръководство за администратора 43](#_Toc193888273)

[7. Авторски права 43](#_Toc193888274)

[III. Заключение 44](#_Toc193888275)

[IV. Списък на използвана литература 46](#_Toc193888276)

[V. Приложения 48](#_Toc193888277)

Оформяне на дипломния проект:

Обем на дипломния проект – минимум 30 страници

Формат: А4; Брой редове в стр.: 30; Брой на знаците: 60 знака в ред

Общ брой на знаците в 1 стр.: 1800 – 2000 знака

Шрифт: Times New Roman, размер 12

Двустранно подравнено

1.5 междуредието

I – 24

1 – 22

* 1. – 20
     1. – 18

1. – 16

I - 14

Текст - 12

# Увод

**Уводът (въведение) съдържа кратко описание на основните цели и очаквани резултати.**

Домашните любимци заемат специално място в живота на хората. Те не само предлагат компания и безусловна обич, но също така допринасят за психическото и физическото благосъстояние на своите стопани. Изследвания показват, че притежаването на домашен любимец може да намали нивата на стрес, да подобри настроението и дори да намали риска от сърдечносъдови заболявания. Освен това, животните често играят роля в терапевтични процеси, подпомагайки хора с различни физически и психически състояния.

Въпреки предимствата на домашните любимци, обстоятелствата кои на техните стопани понякога се променят и те не могат да се грижат повече за тях. В такива случаи намирането на подходящ нов дом за животното става приоритет. Често хората прибягват до социални мрежи или приюти, но тези методи не винаги гарантират най-добрия резултат както за животното, така и за новите му стопани. Ето защо е оправдана нуждата от специализирана платформа, която да улесни доброволната размяна на домашни любимци между хора, които могат да предложат адекватна грижа и подходяща среда.

Настоящият дипломен проект предлага разработването на платформа за доброволна размяна на домашни любимци, която ще свързва собственици на животни, желаещи да ги предадат на друг стопанин, с хора, търсещи нов любимец. Чрез използване на съвременни технологии платформата ще улесни процеса на намиране на най-подходящото място за всяко животно. Основната цел на проекта е да осигури по-безопасна, удобна и ефективна алтернатива за доброволна размяна на домашни любимци, която да минимизира стреса за животните и да осигури най-добрите възможни условия за тяхното отглеждане.

# Основна част

## 1. Цел на ДП.

**Формулира се целта на дипломния проект и задачите, които трябва да бъдат решени, за да се постигне тази цел.**

Основната цел на проекта е да създаде интуитивна и ефективна платформа, която да улесни доброволната размяна на домашни любимци, като осигури безопасен и удобен процес както за настоящите, така и за бъдещите стопани. Чрез системата потребителите ще могат да регистрират своите животни за осиновяване, предоставяйки подробна информация за тях, включително снимки, описание и изисквания за новия стопанин. Платформата ще предлага възможност за лесно управление на профилите и комуникация между заинтересованите страни.

След създаването на профил, потребителят ще има достъп до база данни с всички налични животни, търсещи нов дом в неговия град. Той ще може да търси и сортира предложенията по различни критерии, като тип животно, порода, възраст и специфични нужди на животното, за да намери най-подходящия домашен любимец според своите предпочитания. Интуитивният интерфейс ще улесни взаимодействието и ще позволи на потребителите да филтрират резултатите спрямо своите изисквания и възможности за отглеждане на животно.

Когато потребителят намери животно, което го интересува, той ще може да изпрати запитване до настоящия стопанин. Стопанинът от своя страна ще може да прегледа профила на потенциалния нов собственик, да отговори на въпросите му и да му предостави детайли за контакт. По този начин ще се създаде директна връзка между настоящия и бъдещия стопанин, което ще увеличи шансовете за намиране на най-подходящия дом за всяко животно.

Тази система ще улесни процеса на намиране на нов дом за домашни любимци, като гарантира, че те попадат в ръцете на отговорни и грижовни стопани. Освен това, платформата ще съкрати времето за осиновяване и ще намали натоварването върху приютите за животни, като предлага алтернатива за директна размяна между собственици. Друга ключова функция на платформата ще бъде възможността за обратна връзка и оценка на осиновителите, което ще повиши сигурността и ще гарантира, че животните попадат в добри ръце.

Потребител без регистрация ще има достъп само до основната информация за няколко животни търсещи си дом в неговия град.

Дизайнът на сайта трябва да е адаптивен и да изглежда еднакво добре на различни устройства.

## 2. Проучване на съществуващи приложения.

**Да се започне с проучване, което описва предпоставките за създаване на**

**приложението. Съдържа описание и анализ на съществуващи подобни програмни**

**системи и продукти, като се цитират /с номерация, която се изнася накрая на**

**отделна страница/ съответните литературни източници.**

В България съществуват няколко платформи, насочени към осиновяване и дарения на домашни любимци. Например, сайтът **"Осинови Ме България"** предлага възможност за осиновяване на бездомни животни, като публикува обяви с базова информация за животните, но липсват по-сложни функционалности като филтриране по град, порода и специфични нужди на животните, както и директна комуникация между потребителитe. [Източник 1.](#AdoptMeBg)

Друга иновативна инициатива е платформата **"Pet Buddy"**, която се фокусира върху персонализирана грижа за домашни любимци чрез месечни "кутии" с продукти за отглеждане. Въпреки че тя успешно обслужва сегмента на потребителите, които искат да осигурят качествени продукти за своите животни, платформата не предоставя функция за доброволна размяна на животни, което ограничава нейния обхват за тези, които се нуждаят от нов дом за своите любимци. [Източник 2.](#InvestorBg)

Платформата **"My-PetPal"** е още един пример, който помага на стопани да открият изгубените си животни и да свържат собственици с нови стопани. Въпреки това, тя е предимно насочена към намиране на изгубени животни, а не към целенасочен обмен или осиновяване поради промяна в обстоятелствата на стопанина. [Източник 3.](#PetBuddyBg)

**Сравнение с предложената идея**

Предложената от нас платформа за доброволна размяна на домашни любимци има за цел да запълни пропуските в съществуващите системи. Докато настоящите платформи се фокусират основно върху осиновяването на бездомни животни или предлагането на продукти за грижа, новата система ще предлага възможност за регистрация както на животни, търсещи нов дом, така и на потенциални стопани, които могат да ги намерят лесно чрез детайлно филтриране по град, порода, възраст и пол.

Допълнително, платформата ще осигури директна комуникация между потребителите чрез запитвания и отговори, което значително ще ускори и улесни процеса на осиновяване. Така се постигат два основни цели: подобряване на ефективността на процеса по намиране на нов дом за животните и минимизиране на стреса за тях в преходния период.

**Заключение**

Изследването на съществуващите платформи в България показва, че въпреки наличието на различни системи за осиновяване на домашни любимци, нито една от тях не предлага интегрирано решение, насочено към доброволна размяна с богати функционалности за персонализирано търсене и директна комуникация между стопани. Предложената платформа цели да отговори на тези нужди, като осигури по-цялостен и адаптивен инструмент, който да отговаря на променящите се социални и икономически реалности в страната.

Това проучване обобщава предпоставките за създаване на приложението и подчертава недостатъците на съществуващите системи, като показва как новата платформа може да предложи по-адаптивно и ефективно решение за доброволна размяна на домашни любимци в България.

## 3. Обхват на проекта

**Какво точно ще бъде разработено и какви са ограниченията на системата.**

Проектът цели разработването на интуитивна и функционална онлайн платформа, която да улесни доброволната размяна на домашни любимци в България. Системата ще бъде достъпна чрез интернет и ще функционира като уебсайт, предоставящ централизирано място за публикуване на обяви за животни, които търсят нов дом.

**Основни функционалности:**

**Потребителски профили и автентикация:**  
Потребителите ще могат да се регистрират и управляват своите профили чрез въвеждане на потребителско име и парола. Автентикацията е осъществена изцяло чрез вътрешни механизми и не включва възможност за влизане чрез Google или други профили от трети страни.

**Административен контрол:**  
Платформата ще разполага с отделни администраторски профили, които ще осъществяват контрол върху съдържанието, публикувано от потребителите, както и върху цялостната поддръжка и сигурност на системата.

**Географски обхват:**  
Проектът обхваща всички областни градове в България. Въпреки това, не са включени и всички по-малки населени места, тъй като се предвижда, че потребителите ще посещават най-близкия до тях областен град за да осиновят животно.

**Поддръжка на всички видове домашни любимци:**  
Системата ще поддържа обяви за всички видове животни, включително кучета, котки, малки животни и дори коне. Това осигурява възможност за разнообразен избор на животни според нуждите и предпочитанията на потребителите.

**Ограничения на системата:**

* **Липса на директна комуникация**

Чат функционалности и възможност за директен контакт между потребителите няма да бъдат интегрирани в системата. Това означава, че комуникацията относно обявите ще се осъществява чрез предоставяне на основни данни за контакт, като по-нататъшната комуникация ще бъде осъществена чрез други социални платформи или директно между потребителите.

* **Географска информация само на ниво областен град**

Системата няма опция за избиране на точно географско положение или GPS координати на животното. При публикуване на обява, потребителят трябва да посочи само областния град, в който се намира животното. Това ограничава възможността за по-прецизно филтриране на местоположението, но отразява практическото предположение, че потребителите ще посещават най-близкия до тях областен град за осиновяване.

В обобщение, системата представлява онлайн уебсайт с потребителски и администраторски профили, предназначен за обхващане на всички областни градове в България и поддържащ всички видове домашни любимци, включително коне. Основният акцент е поставен върху сигурността и ефективността на процеса по осиновяване, като социалните аспекти на комуникацията се оставят на външни платформи.

## 4. Технологии

**Описание на използваните технологии, програмни езици,**

**платформи и инструменти. Аргументирано се избират подходящи**

**технологии, като се цитират съответните литературни източници.**

**СПОМЕНИ ЗАЩО ПОЛЗВАШ ТАЗИ ТЕХНОГИЯ В 2-3 изречение**

### 4.1 .NET 8

Технологията .NET 8 представлява последната итерация от универсалната платформа на Microsoft, официално пусната през ноември 2023 г. като издание с дългосрочна поддръжка (LTS). Тази версия носи редица подобрения в производителността, оптимизацията на паметта и интеграцията с облачни услуги, като позволява създаването на високоефективни, кросплатформени приложения, способни да работят стабилно на различни хардуерни архитектури, включително ARM64. Сред функционалностите на .NET 8 са усъвършенствани минимални API, подобрена поддръжка за асинхронно програмиране и по-интуитивен синтаксис за работа с данни чрез LINQ, което значително намалява времето за изпълнение на приложенията. Допълнително, силната интеграция с популярни инструменти за разработка като Visual Studio и Visual Studio Code предоставя на разработчиците богат набор от библиотеки и ресурси, улесняващи бързото и ефективно разработване на софтуер. В сравнение с други езици за програмиране, .NET 8 се отличава със своята стабилност, мащабируемост и богат набор от вградени функции за сигурност, които гарантират надеждност дори при големи, критични приложения. Със своя модерен и доказан подход, .NET 8 предлага значителни предимства пред конкурентните технологии, като осигурява ефективно и мащабируемо решение за съвременната разработка, подкрепено от обширна общност и дългосрочна поддръжка.

### 4.2 Visual Studio 2022

Visual Studio 2022 е най-новата версия на средата за разработка (Integrated Development Environment или IDE), разработена от Microsoft. Тя представлява цялостен набор от инструменти и услуги за разработка, които позволяват на разработчиците да създават широк спектър от приложения, включително настолни, уеб, мобилни, игрови и базирани в облака приложения. Средата поддържа над 36 езика за програмиране, сред които: C#, C++, F#, JavaScript, HTML, CSS, XML, Python и много други, както и множество инструменти като Code Editor и Debugger, Nuget Packages, Git, IntelliCode – (интелигентна система за предложения за дописване на код), инструменти за качествен автоматично генериран код, Server Explorer и други. Visual Studio има както платени версии (Professional и Enterprise), така и безплатна пълна версия (Community).

### 4.3 C#

C# е модерен, обектно-ориентиран език за програмиране, разработен от Microsoft. От началото си през 2002 г. бързо се превръща в един от най-популярните езици за програмиране, използвани за разработване на настолни приложения за Windows, уеб приложения, мобилни приложения, игри и др. C# е силно типизиран език, което означава, че всички променливи трябва да бъдат декларирани с определен тип данни. Важна характеристика е така нареченият garbage-collector, чрез който системата за изпълнение автоматично управлява паметта вместо разработчиците, като освобождава обекти, които вече не са необходими. Във връзка с проекта, C# е подходящ избор заради възможността за писане на асинхронен код, който ще позволи по-добра и по-бърза работа на приложението, което ще доведе и до по-добро потребителско изживяване.

### 4.4 ASP.NET CORE

ASP.NET е web application framework, разработен от Microsoft, който позволява на разработчиците да създават динамични, интерактивни, мащабируеми, надеждни и сигурни уеб приложения и уебсайтове. ASP.NET се основава на .NET Framework и поддържа широк набор от езици за програмиране, включително C#. Поддържа и редица модели за програмиране, включително Web Forms, MVC (Model-View-Controller) и Web API. В комбинация с други решения на Microsoft като SignalR и Identity Core чрез ASP.NЕТ може да се създаде стабилно уеб приложение със система за акаунти и роли на различните потребители. ASP.NЕТ може да се комбинира с front-end framework, различен от тези на Microsoft, като Angular, ReactJS и други.

ASP.NET е една от най-популярните технологии за web development и през 2022г. постига около 37% пазарен дял, като едновременно с това приблизително 32% от уеб приложенията днес използват ASP.NET. Заради предимствата си пред подобни технологии като Django и Angular.js, ASP.NET е подходящ за този проект и ще даде възможност за създаване на Blazor Server проект.

### 4.5 MSSQL Server

MSSQL е вид релационна база данни, базирана на SQL (Structured Query Language). При релационните бази данни група от обекти се пази в таблица, като всеки ред съответства на даден обект, а всяка колона ­– на характеристиките му. MSSQL Server осигурява достъпност, мащабируемост, сигурност и производителност. Той поддържа различни типове данни, включително текст, изображения и XML, и осигурява поддръжка на сложни заявки, обработка на транзакции и съхранение на данни. В комбинация с технология като Entity Framework MSSQL Server ни позволява да запазваме информацията за обектите си в релационна база данни без да се налага да пишем ръчно SQL заявки.

### 4.6 MSSQL Server Management Studio

MSSQL Server Management Studio (SSMS) е инструмент с графичен потребителски интерфейс, който се използва за управление и администриране на бази данни на Microsoft SQL Server. Той позволява на администраторите на бази данни и разработчиците да изпълняват широк кръг от задачи като конфигуриране и наблюдение на сървъри, създаване и управление на бази данни, както и писане и изпълнение на SQL заявки. Той предоставя широк набор от функционални възможности за управление на бази данни на SQL Server, включително: Query Editor, Object Explore, Activity Monitor, Database Backup and Restore, Integration, Analysis и Reporting Services

### 4.7 Entity Framework Core

ORM (Object-Relational Mapping) технологиите позволяват на разработчиците да пишат код на познат обектно-ориентиран език (като C# или Java), за да взаимодействат с базата данни, без да се налага да пишат директно SQL заявки и така позволяват на разработчиците да се съсредоточат върху бизнес логиката на приложението. В този проект се използва Entity Framework Core (EF Core), който е ORM технология, разработена от Microsoft. EF Core позволява на разработчиците да работят с базата данни, използвайки специфични за областта обекти и LINQ заявки. Той поддържа различни доставчици на бази данни, включително Microsoft SQL Server, MySQL, PostgreSQL, SQLite и други. С негова помощ може да се създават приложения чрез два подхода– Code First и Database First.

Проектът е създаден с помощта на EF Core и Code First модела, при който се създават първо класовете, отговарящи на моделите в приложението, след което EF Core създава базата данни заедно с необходимите връзки и ключове по модела, изграден от разработчика с помощта на Data Annotations и наследяване и конфигуриране на DbContext класа.

### 4.8 LINQ

Language-Integrated Query (LINQ) е името на набор от технологии, базирани на интегрирането на възможностите за заявки директно в езика C #. Традиционно заявките срещу данни се изразяват като прости низове без проверка на типа при време на компилация или поддръжка на IntelliSense. Освен това трябва да научите различен език за заявки за всеки тип източник на данни: SQL бази данни, XML документи, различни уеб услуги и т.н. С LINQ заявката е първокласна езикова конструкция, подобно на класове, методи, събития. Пишете заявки срещу силно типизирани колекции от обекти, като използвате ключови думи на език и познати оператори. LINQ семейството от технологии осигурява последователно изпитание за обекти (LINQ to Objects), релационни бази данни (LINQ до SQL) и XML (LINQ до XML). LINQ поддържа два вида синтаксис на писане на заявки (LINQ Query Syntax) и (LINQ Method Syntax).

### 4.9 ASP.NET Core Identity

ASP.NET Core Identity е система за управление на потребители, роли и удостоверяване на потребители в ASP.NET уеб приложения. Тя предоставя оторизация за контрол на достъпа до различни ресурси и персонализиране на поведението на аутентикацията и оторизацията, за да отговаря на специфичните нужди на приложението. ASP.NET Core Identity поддържа различни методи за удостоверяване, включително локални акаунти, социални влизания (например Facebook, Google) и двуфакторно удостоверяване. Включва също и вградена поддръжка за хеширане и валидиране на пароли, което предотвратява често срещани уязвимости в сигурността. В проекта е използван ASP.NET Core Identity, защото е необходимо да се поддържат и установяват ролите на различните потребители в системата и той прави изграждането на структурата от роли и профили на потребителите по-лесна и изчистена.

### 4.10 Уеб браузър

Уеб браузърът е приложение, което позволява на потребителите да имат достъп до и да разглеждат уеб страници в интернет. Уеб браузърите предоставят графичен потребителски интерфейс, чрез който потребителите могат да взаимодействат с уеб съдържание, включително текст, изображения, видеоклипове и друга мултимедия. Достъпът до приложенията на Blazor Server, каквото ще е и използваното за проекта, може да се осъществи чрез всеки съвременен уеб браузър, който поддържа JavaScript. Въпреки това, някои по-стари версии на тези браузъри може да не се поддържат напълно или да имат ограничения в производителността при работа с приложенията на Blazor Server. Най-популярните уеб браузъри включват: Google Chrome, Mozilla Firefox, Apple Safari, Brave, Opera и други.

### 4.11 HTML

HTML (Hypertext Markup Language – език за маркиране на хипертекст) се използва за създаване и структуриране на съдържание за уеб. Това е стандартният език, използван за създаване на уеб страници и други документи, които могат да се разглеждат в уеб браузър.

HTML е език за маркиране, което означава, че използва тагове за определяне на структурата и съдържанието на дадена уеб страница. Тези тагове се използват за създаване на заглавия, параграфи, списъци, връзки, изображения, форми, таблици и други елементи на уеб страницата. Съдържанието на уебстраницата се определя чрез комбинация от HTML тагове и текст.

HTML е в основата на Интернет и е от съществено значение за създаването на уеб страници и други уеб базирани приложения. Той се използва в комбинация с други уеб технологии като CSS и JavaScript за създаване на интерактивни и интересни уеб изживявания.

HTML5 е най-новата версия на HTML, която включва нови функции, като поддръжка на мултимедия, нови семантични елементи и подобрена достъпност. HTML5 включва и API(Application Programming Interface) за геолокация, плъзгане и пускане и други усъвършенствани функции, което улеснява разработчиците при създаването на богати и интерактивни уеб приложения.

### 4.12 CSS

CSS (Cascading Style Sheets) е език, който се използва за описание на представянето на документ, написан в HTML или XML. Той определя как елементите на HTML да се показват на екрана, в печата или в други медии. CSS отделя представянето на документа от неговото съдържание и дава възможност на разработчиците да контролират оформлението, типографията, цветовете и други визуални аспекти на дадена уеб страница.

CSS се използва в комбинация с HTML за създаване на визуално привлекателни и адаптивни уебсайтове. Той позволява на разработчиците да прилагат стилизиране към отделни HTML елементи или към групи от елементи, като използват селектори за насочване към конкретни елементи на страницата. CSS също така позволява създаването на адаптивни дизайни, които регулират оформлението и стила на страницата в зависимост от размера на екрана и ориентацията на устройството, което се използва за разглеждане.

CSS се състои от поредица от правила, които определят стилизирането на дадена страница. Всяко правило се състои от селектор, който насочва към HTML елементите, към които ще се прилага стилизирането, и набор от декларации, които определят свойствата на стилизирането и техните стойности. CSS може да се запише в реда на HTML документа, в отделен файл за стилове или да се вгради в един HTML документ.

### 4.13 Javascript

JavaScript е популярен език за програмиране от високо ниво, който се използва предимно за създаване на динамични и интерактивни уеб страници. Създаден е през 1995 г. от Брендън Айх по време на работата му в Netscape Communications Corporation.

JavaScript е „client side“ скриптов език, което означава, че се изпълнява в уеб браузъра на потребителя, а не на уеб сървър. Често се използва в комбинация с HTML и CSS за добавяне на интерактивност и функционалност към уеб страниците. JavaScript се използва и в „server side“ програмирането, като например в Node.js, популярна среда за изпълнение на JavaScript за изграждане на приложения от страна на сървъра.

JavaScript е обектно-ориентиран език, което означава, че позволява на разработчиците да дефинират обекти и техните свойства и методи. Той също така поддържа парадигми на функционалното програмиране, като например функции от по-висок ред и затваряне.

JavaScript разполага с широк набор от вградени функции и обекти, като масиви, низове и регулярни изрази, които го превръщат в мощен и гъвкав език за уеб разработка. Той също така разполага с голяма екосистема от библиотеки и рамки на трети страни, като React, Angular и Vue.js, които осигуряват допълнителна функционалност и инструменти за изграждане на сложни уеб приложения.

### 4.14 Razor

Razor е синтаксис за създаване на динамични уеб страници в ASP.NET Синтаксисът Razor е синтаксис за маркиране, който комбинира HTML и код от страна на сървъра, позволявайки на разработчиците да създават динамични уеб страници, които могат да се изпълняват на сървъра.

Razor бива въведен в ASP.NET MVC 3 като алтернатива на традиционния синтаксис на ASP.NET Web Forms. Той е проектиран така, че да бъде прост и интуитивен, позволявайки на разработчиците да пишат чист и поддържан код. Синтаксисът Razor поддържа различни езици за програмиране, включително C#, Visual Basic и F#.

Синтаксисът на Razor позволява на разработчиците да вграждат код от страна на сървъра в HTML, като използват символа "@". Този код може да включва C# изрази, оператори и структури за управление, като например if/else оператори и цикли. Синтаксисът на Razor също така предоставя функции като оформления, частични изгледи и компоненти на изгледи, които позволяват на разработчиците да създават многократно използваем и модулен код за своите уеб страници.

Едно от основните предимства на Razor е, че позволява на разработчиците да пишат код от страна на сървъра директно в своите HTML маркировки, което улеснява създаването на динамични уеб страници и избягва нуждата от излишни файлове с код. Razor също така осигурява силна проверка на типовете и други функции за безопасност, които помагат за намаляване на грешките и подобряване на цялостното качество на кода.

### 4.15 Bootstrap Framework

Bootstrap е популярна рамка с отворен код, която се използва за създаване на адаптивни уеб проекти, ориентирани към мобилните устройства. Тя е създадена от разработчиците на Twitter Марк Ото и Джейкъб Торнтън и е пусната през 2011 г.

Bootstrap предоставя колекция от предварително създадени стилове CSS и компоненти JavaScript, които разработчиците могат да използват за бързо и лесно създаване на уеб страници и уеб приложения. Рамката е проектирана така, че да бъде гъвкава и приспособима, което позволява на разработчиците да създават свои собствени уникални дизайни, като същевременно използват основната функционалност на рамката.

Bootstrap включва адаптивна решетъчна система, която позволява на разработчиците да създават оформления, които се адаптират към различни размери на екрана и устройствата. Тя включва също така редица компоненти на потребителския интерфейс, като навигационни менюта, форми, бутони, сигнали и модалности, което улеснява създаването на последователни и визуално привлекателни интерфейси.

Bootstrap е съвместим с всички съвременни уеб браузъри и е проектиран да работи добре с други фронт-енд технологии, като например рамки на JavaScript като React и Angular. Той има голяма и активна общност от разработчици, които допринасят за развитието му и осигуряват поддръжка и ресурси за другите, които го използват.

### 4.16 NUnit

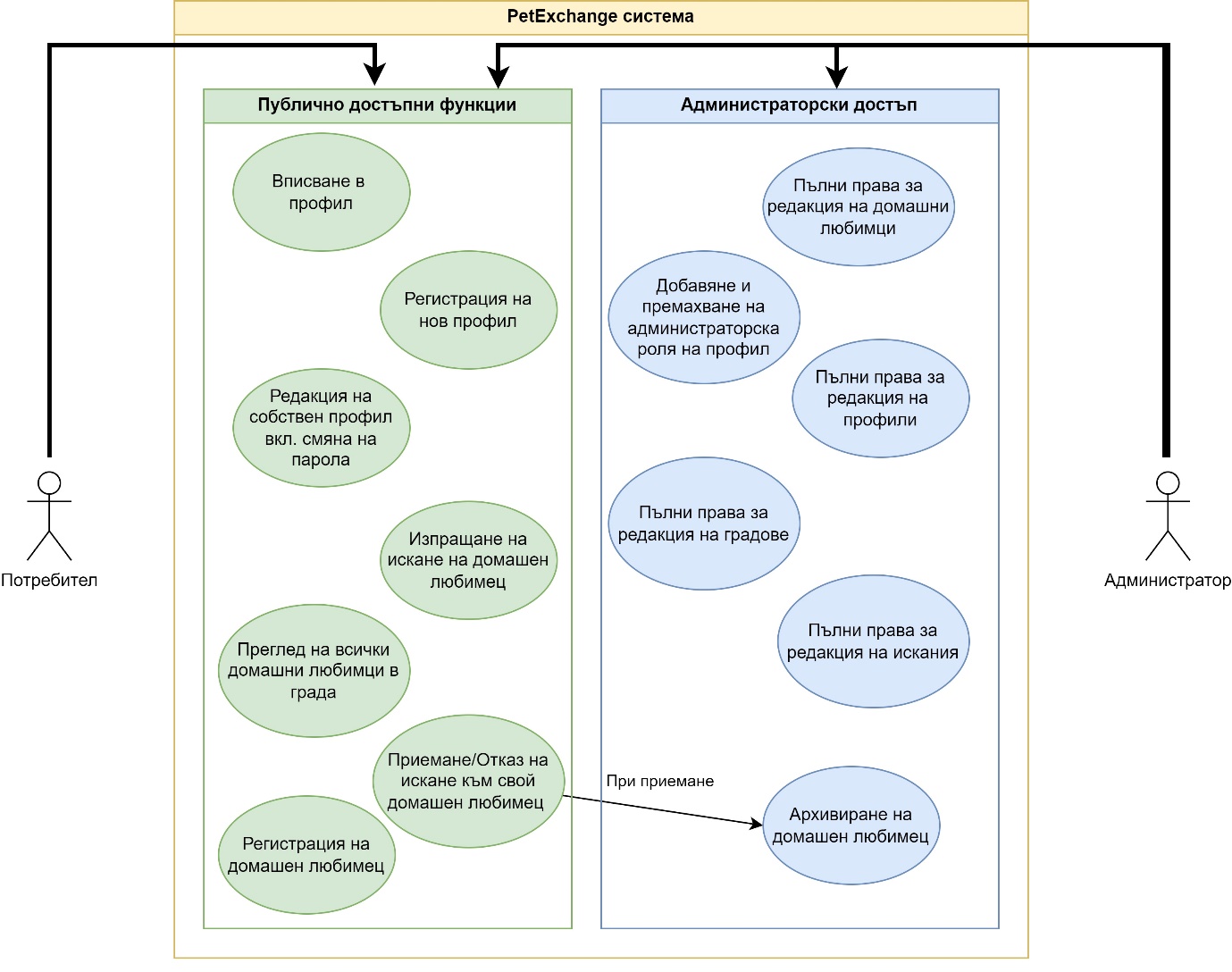
UNIT технологията за тестване представлява ключов компонент в съвременната разработка на софтуер, като се фокусира върху изолираното тестване на отделни функционални единици от кода. Този подход, популяризиран още с пускането на JUnit през 2000 г., е възприет като стандарт в различни програмни езици и среди, като предоставя възможност за бързо откриване на дефекти на ниво модул. С помощта на UNIT технологии разработчиците могат да гарантират, че всяка отделна функция или метод работи правилно преди да се интегрират в цялостното приложение.

Една от основните функционалности на UNIT технологиите е възможността за автоматизация на тестовите случаи, което позволява бърза и надеждна интеграция в CI/CD (Continuous Integration/Continuous Deployment) процесите. Фреймуъркове като NUnit, xUnit и JUnit предлагат богати възможности – от параметризирани тестове до детайлни отчети за грешки и възможност за използване на mocking библиотеки, които улесняват симулацията на зависими компоненти.

Предимствата на UNIT технологиите спрямо други методи за тестване са многобройни. Те предоставят по-прецизни обратни връзки, позволявайки на разработчиците да локализират и отстраняват проблемите в самото начало на разработката, което води до значително намаляване на разходите за поддръжка и подобряване на качеството на кода. В сравнение с интеграционното или системното тестване, UNIT тестовете са по-бързи, по-лесно поддържани и се изпълняват независимо от външни системи, което ги прави незаменими в динамичните среди на Agile и DevOps методологиите.

## 5. Реализация

### 5.1 Функционалност

Изчерпателен списък на всички функции и характеристики, които софтуерът трябва да притежава.

*Фигура X, “Use Case” диаграма на приложението*

Системата **PetExchange** предоставя различни функционалности както за обикновените потребители, така и за администраторите. Те са разделени в две основни категории: **Публично достъпни функции** и **Администраторски достъп**.

**1. Публично достъпни функции (за потребители)**

Обикновените потребители имат достъп до следните функционалности:

**Вписване в профил** – Потребителите могат да влизат в своя акаунт, за да използват пълната функционалност на системата.

**Регистрация на нов профил** – Новите потребители могат да създадат профил в системата.

**Редакция на собствен профил (включително смяна на парола)** – Всеки потребител може да редактира информацията в профила си, включително да променя паролата си.

**Изпращане на искане на домашен любимец** – Потребителите могат да заявяват интерес към определен домашен любимец.

**Преглед на всички домашни любимци в града** – Достъп до списък с наличните домашни любимци в определен град.

**Приемане/Отказ на искане към свой домашен любимец** – Ако потребител притежава регистриран домашен любимец в системата, той може да приема или отказва заявки от други потребители. При приемане, регистрираното животно се архивира и спира да бъде показвано публично.

**Регистрация на домашен любимец** – Възможност за добавяне на домашенлюбимец към системата.

**2. Администраторски достъп**

Администраторите разполагат с допълнителни права, които им позволяват управление на системата:

**Пълни права за редакция на домашни любимци** – Администраторите могат да коригират информацията за всички домашни любимци в системата.

**Добавяне и премахване на администраторска роля на профил** – Администраторите могат да дават или премахват административни права от потребителски профили.

**Пълни права за редакция на профили** – Те могат да редактират потребителски профили, ако е необходимо.

**Пълни права за редакция на градове** – Възможност за управление на списъка с наличните градове в системата.

**Пълни права за редакция на искания** – Контрол върху изпратените искания за домашни любимци.

**Архивиране на домашен любимец** – Администраторът може да архивира домашния любимец, дори и без одобрено искане за него

#### 5.1.1 База Данни

Посочва основната софтуерна система, която управлява

взаимодействията между потребителите и базата данни.

Примери са MySQL, посочена в предишна точка

● Дефинира структурата на базата данни, включваща

таблици, атрибути (колони), типове данни и връзки между

таблиците /посочва графично изображение/.

● Посочва езикът, който се използва за извличане и управление

на данни в базата данни.

● Диаграми: архитектурата на базата данни и взаимодействието

между компонентите (UML диаграми).

UML CLASS DIAGRAM

DATABASE SCREENSHOT

### 5.2 Системни изисквания

Изисквания, свързани с производителност, сигурност, мащабируемост, използваемост и др.

Системните изискванияр хардуер, софтуер

### 5.3 Експлоатация

Потребителските роли за необходими тук

Добави точка, че няма да се иска да се вземе печалба от сайта. Само чрез реклами ще се поддържа хостинга и т.н.

### 5.4 Архитектурен дизайн

За да бъдат изпълнени поставените изискванията (5.1 до 5.3), се реализира следния архитектурен дизайн.

Описание на софтуерната архитектура,

включително структурни компоненти, като слоеве, модули, бази

данни и интерфейси.

За да се реализира желаната функционалност(т. 5.1) е необходимо по-обхватно изпълнение на frontend частта от проекта, което ознавача, че са реализирани над 7 контролера, всеки от които изпълнява конкретна дейност. Модели, които обхващат разнобразието от моделираните активности на приложението.

#### 5.4.1 Data Layer

Основната част от софтуера, който е ангажиран с организацията и управление на данните, необходими за изпълнение на логиката на приложението.

##### Moдели

Класовете “модели” в приложението служат за сформирането на таблици на база на техните полета. Чрез подхода “code-first” и EntityFramework, се създава ORM (Object-Relational Mapping) връзка между моделите и създадените таблици. Имплементирана е валидация на данни чрез анотации в дефиницията на моделите. Допълнителна валидация на данни се прави в слоя ASP.NET MVC (Връзка към този раздел)

Съществуват 4 модела в приложението:

* Pet
* Town
* User
* UserRequest

*Бележка:*

*Pet и User, имат поле PhotoPath пазещо пътят към снимката на съответно домашното животно или потребителя. Изображението се пази в директорията wwwroot в проекта в папки pet или user.*

*Също така Pet и User имат поле IsActive, което определя тяхната видимост в много функции на платформата. При изтриване на модел, той не се премахва от базата данни, но неговата видимост се изключва. Това “псевдо-изтриване” се прави с оглед на мерки за предпазването от случайно изтриване на базата данни при каскада на операцията “изтрий”, както и за да се съхранят данните за бъдещо активиране. Чрез този подход информация никога няма да се изгуби.*

*Полето IsActive на UserRequest, няма такава “псевдо-изтрий” функция. То служи за визуално определяне на статуса в интернет страницата на приложението и за скриване/показване на бутони за изтрий/приеми.*

###### Pet

A screenshot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Модела Pet*

Моделът Pet (Фигура X.) служи за описването на един домашен любимец в базата данни. Той има следните полета:

1. Id, Guid – уникален първичен ключ за таблицата.
2. Name, String – текстово поле за името на домашният любимец.
3. PhotoPath, String – текстово поле пазещо пътят към снимката на домашния любимец.
4. AddedOn, DateTime – поле за дата, служещо за запазването на датата на регистриране на животното в системата.
5. AdoptedOn, DateTime – поле за дата, служещо за запазването на датата на осиновяване на животното в системата. Първоначално стойността на полето е нулевата стойност.
6. Breed, String – текстово поле съдържащо името на породата на животното.
7. Birthday, DateTime – поле за дата, служещо за запазване на рождената дата на животното.
8. AgeDays, Int – поле, което присвоява възрастта на животното в дни чрез пресмятането на разликата между текущата дата и рождението.
9. AgeEnum, PetAgeEnum – поле от изброим тип данни, което пази дали животното се определя като възрастно или младо, според това дали е навършило възраст от 90 дни или не. (Виж [vii. PetAgeEnum](#PetAgeEnum))
10. PetType, PetTypeEnum – поле от изброим тип данни, което пази типа на животното от множеството поддържани типове за домашен любимец. (Виж [v. PetTypeEnum](#PetTypeEnum))
11. Gender, GenderEnum – поле от изброим тип данни, което пази пола на животното. (Виж [viii. GenderEnum](#GenderEnum))
12. Description, String – текстово поле за кратко описание на животното, написано от неговия стопанин.
13. IncludesCage, Bool – поле пазещо логическа стойност дали животното при осиновяване включва и клетка, или е само то.
14. UserId, Guid – поле съдържащо външният ключ към потребител - стопанина на животното.
15. User, User – поле съдържащо обект от класа User, пазещ информация за стопанина. (Виж [iii. User](#User))
16. IsActive, Bool – поле от тип логическа стойност, определящо дали записът е активен.
17. Town, Town – поле съдържащо обект от класа Town, пазещо града, в който се намира животното. Не е външен ключ, а присвоява данните от обекта User.
18. TownId – пази ключа на града. Не е външен ключ, а присвоява данни от обекта User.
19. UserRequests, List<UserRequest> - списък, осъществяващ връзка “един към много” с таблицата от модел UserRequest и пазещ всички искания за животното от потребители на платформата. (Виж [iv. UserRequest](#UserRequest))

###### Town

A screenshot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Модела Town*

Моделът Town (Фигура X.) служи за описването на поддържан град в базата данни. На този етап само всички областни градове в България се поддържат.

Той има следните полета:

1. Id, Guid – уникален първичен ключ за таблицата.
2. Name, String – текстово поле за името на града.

###### User

A screen shot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Модела User*

Моделът User (Фигура Х.) служи за запазването на потребителски, както и администраторски профил в системата. Той наследява класа IdentityUser, използвайки неговите полета и функции за автентикация и оторизация. Също така включва полета за първичен ключ и хеширана парола, които класът User автоматично присвоява. Другите полета на IdenityUser са извън обхвата на този проект и не се използват активно.

Класът User има следните полета:

1. Id, Guid – уникален първичен ключ за таблицата.
2. Еmail, String – текстово поле за адрес на електронна поща на потребителя.
3. UserName, String – текстово поле за потребителското име на потребителя, което служи за автентикация.
4. PhoneNumber, String – текстове поле за телефонния номер на потребителя.
5. Name, String – текстове поле за името на потребителя
6. PhotoPath, String – текстове поле, пазещо пътя към файла на профилната снимка, която се показва в платформата.
7. IsActive, Bool – поле от тип логическа стойност, определящо дали записът е активен.
8. Role, RoleEnum - поле от изброим тип, определящо ролята на акаунта. (Виж [vi. RoleEnum](#RoleEnum))
9. TownId, Guid - поле за външният ключ към таблицата за градовете.
10. Town, Town - поле, пазещо обекта от тип град свързан с профила. (Виж [ii. Town](#Town))
11. Pets, List<Pet> - списък, осъществяващ връзка “един към много” с таблицата от модел Pet и пазещ всички регистрирани животни от потребителя. (Виж [i. Pet](#Pet))
12. RequestOutbox, List<UserRequest> - списък, осъществяващ връзка “един към много” с таблицата от модел UserRequest и пазещ всички изпратени искания към чужди животни от потребителя. (Виж [iv. UserRequest](#UserRequest))
13. RequestInbox, List<UserRequest> - списък, осъществяващ връзка “един към много” с таблицата от модел UserRequest и пазещ всички получени искания, изпратени към животни, на които потребителят е стопанин.

###### UserRequest

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура X. Модела UserRequest*

Моделът UserRequest (Фигура Х.) служи за запазване на искане на домашен любимец в базата данни.

Той има следните полета:

1. Id, Guid – уникален първичен ключ за таблицата.
2. CreatedOn, DateTime – поле за дата, пазещо кога искането е било създадено.
3. DeniedOn, DateTime – поле за дата, пазещо кога искането е било отказано от потребителя получател. Първоначалната му стойност е нулевата стойност(“null”).
4. CanceledOn, DateTime – поле за дата, пазещо кога искането е било отхвърлено от потребителя, който първоначално го е създал. Първоначалната му стойност е null.
5. AcceptedOn, DateTime – поле за дата, пазещо кога искането е било прието от потребителя получател. Първоначалната му стойност е null.
6. SenderId, Guid – поле за външен ключ към таблицата на модела User. То пази връзката към потребителя, който е създал искането.
7. Sender, User – поле пазещо обект от тип User – това е потребителят “изпращач”. (Виж [iii. User](#User))
8. RecipientId, Guid – поле за външен ключ към таблица на модела User – това е потребителят получил искането.
9. Recipient, User – поле пазещо обект от тип User – това е потребителят “получател”.
10. PetId, Guid – поле за външен ключ към таблицата на модела Pet – това е домашният любимец, за когото е направено искането.
11. Pet, Pet – поле пазещо обект от тип Pet – това е исканият домашен любимец.
12. RequestMessage, String – текстово поле, пазещо първоначалното съобщение, което получателят ще получи относно искането му на домашния любимец.
13. AnswerMessage, String – текстово поле, пазещо съобщението, което получателят ще изпрати на изпращача. Показва се както при отказ, така и при приемане на искането.
14. IsActive, Bool – логическо поле, което определя дали искането е активно още или не. При получаване на отказ или прием от получателя или отхвърляне от изпращача, искането става неактивно.

##### Помощни класове

В този раздел са разгледани помощни класове, които покриват различни функционалности, допълващи главните модели.

###### PetTypeEnum

A screenshot of a computer code

AI-generated content may be incorrect.

*Фигура X. Изброимият тип PetTypeEnum*

Изброимият тип PetTypeEnum (Фигура Х.) съдържа всички типове домашни любимци, които платформата поддържа. Те са: котки, кучета, риби, малки бозайници, птици, влечуги, земноводни, коне и други.

###### RoleEnum

A screenshot of a computer code

AI-generated content may be incorrect.

*Фигура X. Изброимият тип RoleEnum*

Изброимият тип RoleEnum (Фигура Х.) съдържа всички роли, които платформата поддържа за оторизация – потребител и администратор.

###### PetAgeEnum

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура X. Изброимият тип PetAgeEnum*

Изброимият тип PetAgeEnum (Фигура Х.) съдържа двете възможности за възраст на животното, с които то е категоризирано – младо или възрастно. (Виж[. i.Pet](#Pet))

###### GenderEnum

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура X. Модела GenderEnum*

Този изброим тип данни(Фигура X.) служи за правилното задаване на пола на всеки домашен любимец. Вариантите са: мъжки, женски или “не се знае”, поради невъзможно определяне на пола при новородени животни или други причини.

###### SelectOption

A screenshot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Класа SelectOption*

Класът SelectOption (Фигура Х.) служи за изграждането на опциите в множество избирателни списъци, които се използват в интернет страницата на приложението.

Той има следните полета:

1. Label, String – текстово поле за видимият етикет на опцията
2. Value, String – текстово поле за стойността, която ще се подаде, ако опцията е избрана
3. Selected, Bool – логическо поле за определяне, дали опцията е избрана по-подразбиране. Това става често при зареждане на стара информация, която вече потребителя е вече попълнил.

###### EnumExtensions

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Метода ToDescriptionString в класа EnumExtensions*

Класът ЕnumExtensions (Фигура Х.) цели да увеличи функционалностите на изброимия тип данни. Чрез метода ToDescriptionString(), който се съдържа в него, се достъпва атрибута за описание на всяка опция. По този начин лесно може да се използва българският превод на всяка опция във всеки дефиниран изброим тип в графическият потребителски интерфейс на интернет страницата.

##### DbContext класовете и интерфейси

Класовете “DbContexts” отговарят за работата с данните в приложението и записите в базата данни. Всяка таблица си има направен отделен dbContext. В тези класове се реализират асинхронно функциите на CRUD (Create, Read, Update, Delete) – създай, прочети, актуализирай, изтрий. Асинхронното изпълнение на тези методи е от изключителна важност, позволяващо множество едновременни заявки от много потребители към базата данни. Това се гарантира чрез 2 интерфейса, които също ще бъдат разгледани в този раздел.

В тези класове има и помощни функции, които служат за извличането на специфични данни от базата данни чрез филтри или други критерии.

*Бележка: DbContext класовете могат и да ползват навигационни свойства при извличането на данни. Тези “свойства” определят дали да се зареди и информацията във външните обекти, свързани с текущия. Чрез логически параметър се определя дали това допълнително зареждане ще се случи или обектите да си останат с нулева стойност.*

###### IdbWithNav

A computer code with blue and white text

AI-generated content may be incorrect.

*Фигура Х. Интерфейса IdbWithNav*

Интерфейсът IdbWithNav (Фигура Х.) служи за определянето на стандарт за функционалностите на един dbContext клас, който има навигационни свойства и може да ги достъпи при извличане на данни.

Интерфейсът е шаблонен клас с два параметъра от неизвестен тип, които ще се инициализират в последствие. Те са:

1. T – типа данни на модела, с който ще се работи
2. K – типа данни на първичния ключ на таблицата на модела. (В нашият случай това винаги ще е Guid)

Интерфейсът гарантира, че следните методи ще бъдат създадени:

1. CreateAsync (T entity), Task –запазване на обект T в базата данни
2. CreateAsync (List<T> entities), Task – запазване на списък от обекти Т в базата данни
3. ReadAsync (K id, bool useNavigationalProperties = false, bool isReadOnly = true), Task<T> - извличане от базата данни запис с ключ К. Съществуват опциите да се извличат и свързаните с записа навигационни свойства, както и дали данните да само за четене. Тези опции са обозначени с два логически параметъра.
4. ReadAllAsync (bool useNavigationalProperties = false, bool isReadOnly = true), Task<List<T>> - наподобява метода ReadAsync, но извличане всички записи от конкретна таблица в базата данни.
5. UpdateAsync (T entity, bool useNavigationalProperties = false), Task - актуализиране на данните на един запис. Има опция за актуализиране и на навигационните свойства, свързани със записа.
6. DeleteAsync (K id) – премахване на запис от таблицата според неговия първичен ключ К. В повечето dbContext класове имплементираното изтриване не е окончателно, а е “псевдо-изтриване”. (Виж [бележката след а) Модели](#БележкатаНаМодели))

###### IdbWithoutNav

A screenshot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Интерфейса IdbWithoutNav*

Интерфейсът IdbWithoutNav (Фигура Х.) наподобява IdbWithNav с разликата, че функционалността да се извличат навигационните свойства не е зададена. Този интерфейс e направен за dbContext класа на модела Town, защото той няма външни ключове. (Виж [xi. IdbWithNav](#IdbWithNav))

###### PetExchangeDbContext

A screenshot of a computer program

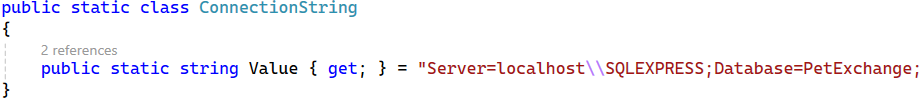
AI-generated content may be incorrect.

*Фигура Х. Класът PetExchangeDbContext*

Класът PetExchangeDbContext (Фигура Х.) служи за връзката с базата данни и определянето на таблиците, които платформата ще използва. Това се случва с дефиницията на 4 DbSet полета (съответно за 4-те модели). Низът за връзка с базата данни e изведен в отделен клас ConnectionString (Виж [xiv. ConnectionString](#ConnectionString)).

Също така този клас съдържа и метод за попълване на примерни записи в базата данни, когато тя е празна. Тези записи ще се използват при представянето на дипломната работа, за да се покажат всички функционалности на проекта.

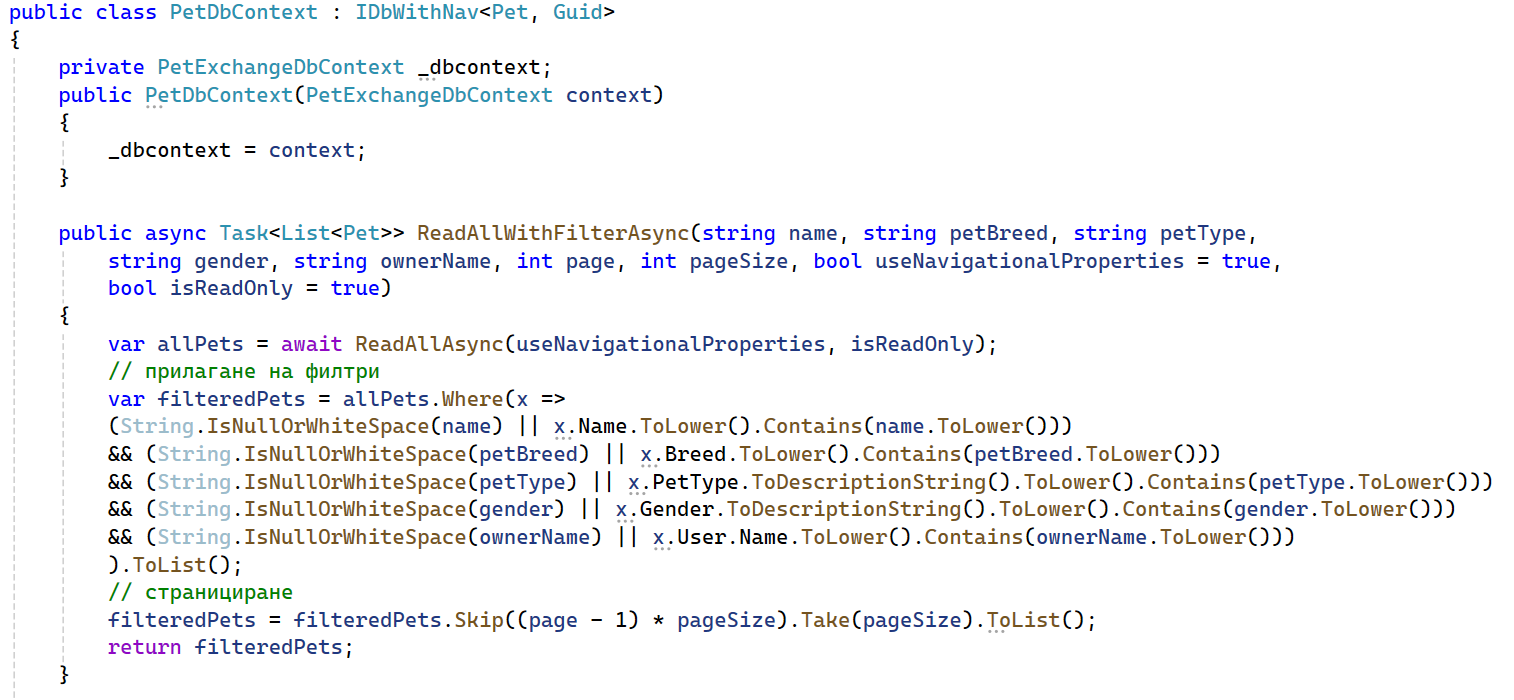
###### ConnectionString



*Фигура Х. Класа ConnectionString*

Низът за връзка с базата данни се отделя в друг клас (Фигура Х.) с оглед мерки на сигурност и лесна четимост. Това също позволява кооперативна работа на много програмисти, като всеки използва свой различен низ за връзка.

###### PetDbContext



*Фигура Х. Класа PetDbContext*

Класът PetDbContext (Фигура Х.) служи за работа със записите на домашни любимци в платформата. Той имплементира интерфейса IdbWithNav и има няколко метода за филтрирано извличане на данни. Тези методи използват LINQ заявки за извличане на нужната информация.

Първият метод за извличане е ReadAllWithFilterAsync (Фигура Х.). Служи за филтриране на домашните любимци при работа с таблици в графичния потребителски интерфейс(Връзка към раздела за таблици). Приемайки няколко параметъра, обозначаващи различните критерии за филтриране(име, порода, тип на животното, пол, име на стопанина), както и параметри свързани със страницирането, навигационните свойства и режима “само четене”, методът връща списък с обектите, които отговарят на критериите. Алгоритимът първо извлича всички записи с метода ReadAllAsync() и след това чрез LINQ заявки проверя всеки критерий първо дали е зададен и после го прилага към списъка. Страцинирането идва след това - според размера на страниците и номерът на текущата страница се взимат данните за текущата страница.

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Метода ReadAllWithFilterAsyncOfUser*

Вторият метод за извличане е ReadAllWithFilterAsyncOfUser (Фигура Х.). Той е аналогичен с предишния, но добавя още 2 критерия – стопанин и активност (“видимост”) на записа. Той се използва за извличането само на домашни любимци, на които потребителят е стопанин и не са “псевдо-изтрити”.

A screenshot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Метода ReadWithFiltersAsync*

Третият метод за извличане е ReadWithFiltersAsync (Фигура Х.). Той се използва в страница “Домашни любимци” (Връзка към тази страница, ако има такъв раздел) за извличане на всички активни неосиновени животни според множество филтри. Параметрите тук на филтрите се задават с листове от дефинираните изброими типове.

A computer screen shot of a code

AI-generated content may be incorrect.

*Фигура Х. Методите Read4NewestAsync() и Read4OldestAsync()*

Следват още 2 метода, които са сходни помежду си – Read4NewestAsync и Read4OldestAsync (Фигура Х.). Те извличат от базата данни съответно 4-те най-скорошно регистрирани животни и 4-те животни, които чакат осиновяване от най-много време. Методите са използвани за извличане на животни, които ще се покажат на главната страница на платформата. (Връзка с главната страница)

Следват методи, свързани с имплементацията на интерфейса за CRUD.

1. **Запазване**

A computer screen shot of text

AI-generated content may be incorrect.

*Фигура Х. Методите за запазване в PetDbContext*

Съществуват 2 метода за създаване на записи в базата данни – CreateAsync(Pet entity) и CreateAsync(List<Pet> pets). Вторият извиква първият за всеки елемент в списъка с обекти, който получава като параметър.

Запазването се случва, като се достъпи DbSet Pets от инстанция на PetExchangeDbContext (Виж [xii. PetExchangeDbContext](#PetExchangeDbContext)), добавя се елемент към него и се запазят промените.

1. **Четене**

A computer screen shot of a program

AI-generated content may be incorrect.

*Фигура Х. Методите за четене в PetDbContext*

Методите за четене ReadAsync и ReadAllAsync (Фигура Х.) съдържат параметри за навигационни свойства и за режим “само четене”.

Първата стъпка на алгоритъма е достъпване на пълния списък с животни чрез DbSet Pets. Втората е, ако е зададено, да се настройват данните да не се следят за промени от EntityFramework, поставейки ги в режим “само четене”. Това става чрез метода AsNoTrackingWithIdentityResolution().

Третата стъпка е зареждането на навигационните свойства, отново при зададена логическа стойност “истина“ на параметъра useNavigationalProperties. Това става чрез множество LINQ заявки (Include и ThenInclude). В зависимост от това колко навигационни свойства съществуват тези заявки може да бъдат повторени няколко пъти.

Накрая в случая на ReadAsync() се връща елемента от списъка с избран първичен ключ или в случая на ReadAllAsync() – целият списък.

1. **Актуализация**

**A screen shot of a computer code

AI-generated content may be incorrect.**

*Фигура Х. Метода за актуализация в PetDbContext*

Методът за актуализация UpdateAsync (Фигура Х) използва метода ReadAsync, за да достъпи нужният запис от базата данни. След това, ако се използват навигационни свойства, се актуализират тези записи чрез метода Update(). Иначе, се актуализира само достъпеният запис.

1. **Изтриване**

**A computer code with text

AI-generated content may be incorrect.**

*Фигура Х. Метода за изтриване в PetDbContext*

Методът за изтриване DeleteAsync (Фигура Х.) “псевдо-изтрива” запис по зададен първичен ключ, като първо го достъпва чрез ReadAsync.

###### TownDbContext

A screenshot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Класа TownDbContext*

Класът TownDbContext (Фигура Х.) наследява IdbWithoutNav, защото класа Town няма външни ключове. По аналогичен начин както PetDbContext, имплементира CRUD, като пропуска функционалностите за навигационните свойства. (Виж [xv. PetDbContext](#PetDbContext))

A computer code with text

AI-generated content may be incorrect.

*Фигура Х. Метода за изтриване в TownDbContext*

Единствената разлика е, че при метода за изтриване (Фигура Х.) окончателно се изтрива записът от базата данни - липсва функция “псевдо-изтриване”.

###### UserDbContext

A screenshot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Класа UserDbContext*

Класът UserDbContext (Фигура Х.) имплементира интерфейсът IdbWithNav и аналогогично като PetDbContext изгражда методи за филтрирано извличане на записи и CRUD. (Виж [xv. PetDbContext](#PetDbContext))

Разликите между двата класа са следните:

1. UserDbContext имплементира метода ChangePassWord (Фигура Х.), който, използвайки вграденият клас userManager<User>, променя паролата на потребител с нова, която отново е хеширана и само нейният хеш е запазен в базата данни.
2. User моделът разчита на задаването на роли за оторизация. Те се слагат и премахват чрез методите на класа userManager - AddToRoleAsync и RemoveFromRoleAsync. (Фигура Х.)

A screenshot of a computer screen

AI-generated content may be incorrect.

*Фигура Х. Добавяне и премахване на роли с UserManager при метода за актуализация на UserDbContext*

###### UserRequestsDbContext

A screenshot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Класа UserRequestsDbContext*

Класът UserRequestsDbContext (Фигура Х.) имплементира интерфейсът IdbWithNav и аналогогично като PetDbContext изгражда методи за филтрирано извличане на записи и CRUD. (Виж [xv. PetDbContext](#PetDbContext))

Класът съдържа и още няколко специализирани методи за работа с искания:

1. **Извличане на входящите и изходящите искания за даден потребител**

A screen shot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Методите за Outbox и Inbox в UserRequestDbContext*

Методите ReadUserRequestOutboxAsync и ReadUserRequestInboxAsync (Фигура Х.) извличат всички записи за изходящи или входящи искания на даден потребител. Филтрирането се извършва с помощта на LINQ заявки, които се прилагат към пълният списък искания от базата данни

1. **Методи за прием/отказ/отхвърляне на искане**

**A screen shot of a computer code

AI-generated content may be incorrect.**

*Фигура Х. Метода CancelAsync на класа UserRequestsDbContext*

Методът CancelAsync (Фигура Х) служи за отхвърляне на искане към домашен любимец от потребителя, който първоначално го е направил. Това се случва с извличане от на запис чрез първичния му ключ, задаване на дата на отхвърляне и запазване на промените.

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Метода AcceptAsync на класа UserRequestsDbContext*

Методът AcceptAsync (Фигура Х) служи за приемане на искане към домашен любимец от потребителя получател. Това се случва с извличане на запис чрез първичния му ключ, задаване на дата на приемане, както и съобщение за отговор и дата на осиновяване, и запазване на промените.

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Метода DenyAsync на класа UserRequestsDbContext*

Методът DenyAsync(Фигура Х) служи за отказ на искане към домашен любимец от потребителя получател. Това се случва с извличане на запис чрез първичния му ключ, задаване на дата на отказване, както и съобщение за отговор, и запазване на промените.

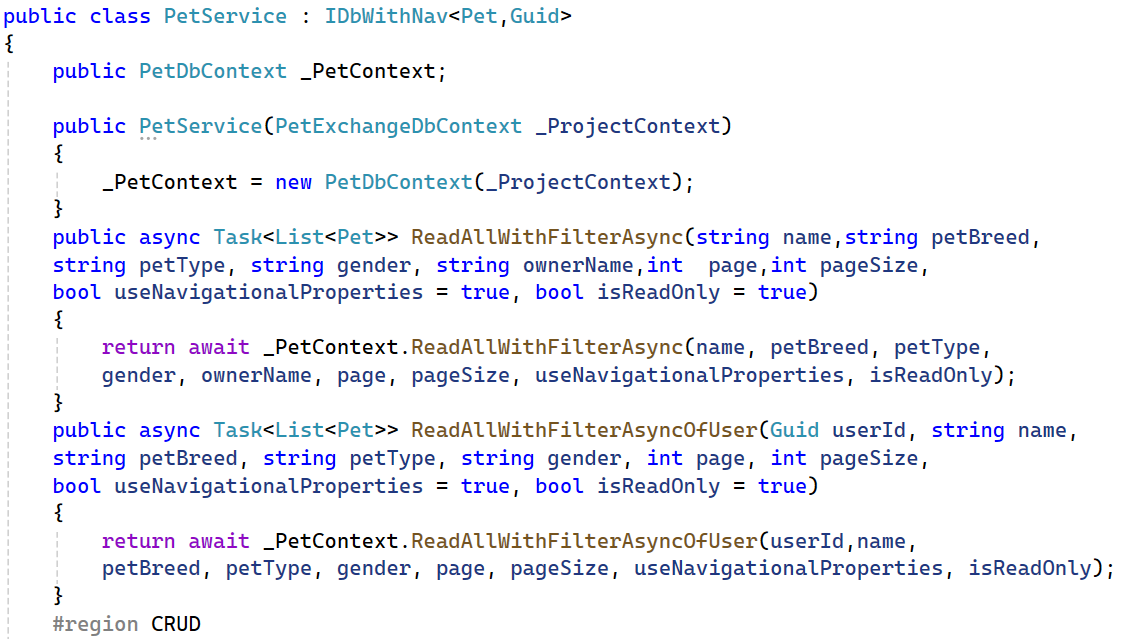
#### 5.4.2 Business Layer

Бизнес слоят е основна част от архитектурата на приложението, която осигурява връзката между слоя за данни (Data Layer) и презентационния слой (Presentation Layer). Неговата роля е:

1. Да осигурява абстракция, като скрива детайлите на достъпа до база данни от потребителския интерфейс.
2. Да следва структурата на методите, дефинирани в DbContext класовете, като извиква методите, създадени в слоя за данни.

Тази архитектура осигурява модулност, лесна поддръжка и повторно използване на кода. Слоят съдържа четири класа – един за всеки дефиниран DbContext клас.

###### PetService



*Фигура Х. Класа PetService*

Класът PetService (Фигура Х.) съдържа методи-дубликати на тези в PetDbContext и ги извиква. (Виж [xv. PetDbContext](#PetDbContext))

###### TownService

A screenshot of a computer program

AI-generated content may be incorrect.

*Фигура Х. Класа TownService*

Класът TownService (Фигура Х.) съдържа методи-дубликати на тези в TownDbContext и ги извиква. (Виж [xvi. TownDbContext](#_TownDbContext))

Също така съдържа и метод GetTownOptions (Фигура Х), който извлича всички регистрирани до момента градове и ги предава под формата на списък от тип SelectOption. Този списък след това се използва в регистрационната форма на приложението. (Виж [ix. SelectOption](#_SelectOption))

###### UserService

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Класа UserService*

Класът UserService (Фигура Х.) съдържа методи-дубликати на тези в UserDbContext и ги извиква. (Виж [xvii. UserDbContext](#_UserDbContext))

###### UserRequestsService

A screen shot of a computer code

AI-generated content may be incorrect.

*Фигура Х. Класа UserRequestsService*

Класът UserRequestsService (Фигура Х.) съдържа методи-дубликати на тези в UserRequestsDbContext и ги извиква. (Виж [xviii. UserRequestsDbContext](#_UserRequestsDbContext))

#### 5.4.3 ASP.NET MVC (Presentation Layer)

Този слой отговаря за графичния интерфейс на платформата и начина, по който потребителите взаимодействат с приложението. Той е изграден с технологията ASP.NET MVC, която следва Model-View-Controller (MVC) архитектурния шаблон.

MVC разделя слоя на три основни компонента:

1. Модели (Models) – представят данните и бизнес логиката на приложението.
2. Изгледи (Views) – визуализират информацията и оформят потребителския интерфейс.
3. *Контролери (Controllers*) – управляват заявките на потребителя, обработват входните данни и координират взаимодействието между моделите и изгледите.

Тази структура улеснява поддръжката, подобрява организацията на кода и позволява по-ясно разделение на отговорностите.

В този раздел ще разгледаме как тези компоненти работят заедно.

##### Модели

##### Изгледи

##### Контролери

###### Program.cs

Ще разгледам по-добро следните главните функционалности:

Контролери, , карта за животни, breadcrumb, таблици и снимки, страници за грешки, филтри,

#### 5.4.4 Test Layer

Приложението се тества с мениджър клас, която се компилира и извършва преди всеки тест. Служи за зареждането на тестовата база данни (в паметта) и други методи за приготовление.

Компонентно тестване на поне една

структурна единица. Резултати от тестовете: Доклад за проведените тестове, включително грешки и корекции. Грешки при круда

За тестване на фронтенд съм ползвам ръчни тестове

###### Пример за тестов мениджър

###### Пример за тест

#### 5.4.5 Осигуряване на сигурност

Спомени атаките

Описание на всички мерки за сигурност

Сигурност на данните

Валидация на данни

Сигурност на контролерите и извеждането от тях данни

Потребителски роли

Автентикация

Ауторизация

Хеширане на пароли

## 6. Потребителска документация

### 6.1 Ръководство за потребителя

Описание на това как

потребителите могат да използват системата. Това може да

включва инструкции за инсталиране, работа с потребителския

интерфейс и справяне с често срещани проблеми.

Виж приложение

### 6.2 Ръководство за администратора

Ръководство за администратора:

Описание на това как системата

се администрира и поддържа, включително настройки,

конфигурации и управление на базата данни.

## 7. Авторски права

Информация за собствеността и правата върху кода и

използваните библиотеки.

# Заключение

Заключението съдържа изводи и предложения за доразвиване на проекта и

възможностите за неговото приложение.

Обобщава постигнатото в дипломния проект, представя възможностите за бъдещо

развитие и усъвършенстване, както и възможностите за приложение на създаденото

приложение.

ПОНЕ 2 СТРАНИЦИ,

ВСИЧКО от-до трудности и постигнатите ползи, тестване

# Списък на използвана литература

Списъкът с използваната литература включва цитираната и използвана в

записката на дипломния проект литература. Започва на отделна страница от основния

текст. При имената на авторите първо се изписва фамилията. Всички описания в

списъка с използваните източници трябва да са подредени по азбучен ред според

фамилията на първия автор на всяка публикация.

**Източници използвани за реализирането на проекта**

* <https://stackoverflow.com>
* <https://www.w3schools.com>
* <https://www.wikipedia.org>
* <https://www.youtube.com>
* <https://getbootstrap.com>
* <https://openai.com>
* <https://www.codecademy.com/learn/learn-c-sharp>
* <https://html.w3school.bg>

1. Учебник по информационни технологии 12 клас, модул 3, Уеб дизайн, издателство Домино
2. Учебник по Информатика за 12. клас – профилирана подготовка МОДУЛ 4: Програмиране на информационни системи, издателство Изкуства

**Източници използвани за проучването**

1. [adoptmebg.com](https://adoptmebg.com/" \t "_blank) – "Осинови Ме България", информация за осиновяване на бездомни животни.

1. [petbuddy.bg](https://petbuddy.bg/kupuvane-ili-osinoviavane-domashen-liubimec-kak-da-izberem/" \t "_blank) – "Купуване или осиновяване на домашен любимец – какво да изберем?", Pet Buddy.

1. [investor.bg](https://www.investor.bg/a/571-it-i-telekomunikatsii/389919-nova-platforma-pomaga-na-stopani-da-otkriyat-domashnite-si-lyubimtsi" \t "_blank) – "Нова платформа помага на стопани да открият домашните си любимци", Investor.bg.

# Приложения

Да са номерирани (Приложение 1, Приложение 2 ...). Приложенията съдържат

документация, която не е намерила място в текста поради ограниченията в обема й

или за по-добра прегледност и подредба. В текста трябва да има препратка към всяко

приложение.

Вмъкнатите изображения, фигури, диаграми и други, в документацията, да бъдат

подходящо форматиране, като разстоянието на текста преди и след тях да бъде не

повече от 6 пункта. Да няма едно изображение на цяла страница.

Notes

● Коментари в кода: Добра практика е да има смислени коментари

в кода, които обясняват какво правят различните части от

програмата.