ЧАСТНА ПРОФЕСИОНАЛНА ГИМНАЗИЯ ЗА ДИГИТАЛНИ НАУКИ „СОФТУНИ БУДИТЕЛ“, гр. София

**ДИПЛОМЕН ПРОЕКТ**

на Вероника Иванова Желева

ученик/ученичка от XII Б клас

професия- код:  **481030**, **“Приложен програмист”**

специалност-код: **4810301**,

**“Приложно програмиране”**

**Тема: Разработване на уеб приложение за намиране на ветеринарен лекар.**

**Ръководител-консултант:** .Кирил Поповски.

Сесия: май-юни 2025г.

Съдържание

[Увод 2](#_Toc196907391)

[Нужда от приложението 3](#_Toc196907392)

[Цел 3](#_Toc196907393)

[Потребители 3](#_Toc196907394)

[Описание 4](#_Toc196907395)

[Същинска част 4](#_Toc196907396)

[Глава 1: Проучвателна част 4](#_Toc196907397)

[1. Преглед на съществуващи решения 4](#_Toc196907398)

[2. Анализ на проблемите и пропуските 6](#_Toc196907399)

[2.1 Липса на цялостно решение 6](#_Toc196907400)

[2.2 Ограничена персонализация 6](#_Toc196907401)

[2.3 Липса на интеграция с други услуги 6](#_Toc196907402)

[Подробен анализ на функционалностите на приложението VetTime 6](#_Toc196907403)

[1. Записване на часове при ветеринар 6](#_Toc196907404)

[2. Регистрация и управление на потребители 7](#_Toc196907405)

[3. Управление на специализации и градове 7](#_Toc196907406)

[4. Администрация на данни и база 7](#_Toc196907407)

[5. Оценки и отзиви 8](#_Toc196907408)

[6. Визуално представяне и UX 8](#_Toc196907409)

[Глава 2: Задание за софтуерна разработка 8](#_Toc196907410)

[1. Функционални изисквания 8](#_Toc196907411)

[1.1 Регистрация на ветеринари 8](#_Toc196907412)

[1.2 Регистрация на клиенти 9](#_Toc196907413)

[1.3 Търсене и препоръки 9](#_Toc196907414)

[1.4 Детайли за ветеринар 9](#_Toc196907415)

[1.5 Резервации 9](#_Toc196907416)

[1.6 Администраторски панел 9](#_Toc196907417)

[2. Нефункционални изисквания 10](#_Toc196907418)

[2.1 Сигурност 10](#_Toc196907419)

[2.2 Производителност 10](#_Toc196907420)

[2.3 Надеждност и поддръжка 10](#_Toc196907421)

[3. Избор на технологии 10](#_Toc196907422)

[4. Моделиране на системата 10](#_Toc196907423)

[4.1 Основни Entities 10](#_Toc196907424)

[4.2 Връзки между основните модели 11](#_Toc196907425)

[5. Представяне (Views) 11](#_Toc196907426)

[5.1 Home 11](#_Toc196907427)

[5.2 Authentication 11](#_Toc196907428)

[6. ViewModels 11](#_Toc196907429)

[7. Контролери 12](#_Toc196907430)

[Трета глава: 13](#_Toc196907431)

[Проектиране, имплементация, тестване и внедряване 13](#_Toc196907432)

[3.1 Проектиране на системата 13](#_Toc196907433)

[3.2 Архитектура на приложението (MVC) 13](#_Toc196907434)

[3.3 Моделиране на данните (словесна ER диаграма) 15](#_Toc196907435)

[Основни таблици и връзки: 15](#_Toc196907436)

[15](#_Toc196907437)

[3.4 Основни действия в приложението 16](#_Toc196907438)

[Регистрация и влизане 16](#_Toc196907439)

[Търсене на ветеринари 18](#_Toc196907440)

[Преглед на детайли и резервация на час (figure 7) 20](#_Toc196907441)

[Добавяне и управление на ветеринарен профил 21](#_Toc196907442)

[3.5 Имплементация и използвани технологии 21](#_Toc196907443)

[3.6 Тестване 21](#_Toc196907444)

[3.7 Внедряване 22](#_Toc196907445)

[**База данни:** 22](#_Toc196907446)

[**Готовност за бъдещи надграждания:** 22](#_Toc196907447)

# 

# Увод

Приложенията за записване на часове са от съществено значение в съвременния свят, особено в области като ветеринарната медицина, където ефективното управление на времето и ресурсите е ключово за предоставянето на качествени услуги. VetTime е уеб приложение, разработено с ASP.NET MVC и SQL Server, което улеснява връзката между клиенти и ветеринарни лекари. То позволява на потребителите да се регистрират като клиенти или ветеринари, да търсят ветеринари по различни критерии (име, град, специализация) и да резервират часове за консултации. Приложението поддържа работно време от понеделник до петък, от 9:00 до 17:00, и предоставя функционалности като филтриране на ветеринари, преглед на детайли и рейтинг, както и управление на профили.

VetTime адресира реални проблеми като липсата на централизирана платформа за ветеринарни услуги, трудностите при намирането на подходящ специалист и необходимостта от лесно и бързо записване на часове. Чрез автоматизация и интуитивен интерфейс, приложението подобрява достъпа до ветеринарни грижи и оптимизира работния процес на ветеринарите. Тази документация разглежда кода на приложението, описвайки всеки клас, метод и връзка, за да се покаже как системата функционира и защо избраните технологии и архитектура са подходящи.

## Нужда от приложението

В забързаната клинична среда медицинските специалисти и пациентите очакват изчистен интерфейс, който ускорява административните процеси, осигурява прозрачност и намалява бюрокрацията. Приложението консолидира данни за прегледи, рецепти и лекарства, елиминира дублирането и подпомага вземането на информирани решения.

## Цел

Основната цел е създаване на уеб приложение, което: позволява лесно записване на часове, поддържа електронни медицински досиета и мащабира безпроблемно.

## Потребители

Това уеб приложение е предназначено да обслужва:

Клиенти-Собственици на домашни лзбимци.

Ветеринарни лекари

## Описание

VetTime е уеб приложение, което използва ASP.NET MVC за създаване на потребителски интерфейс, Entity Framework Core за управление на базата данни и SQL Server за съхранение на данни. Приложението поддържа следните основни функционалности:

- Регистрация и влизане : Потребителите могат да се регистрират като клиенти или ветеринари, като предоставят лична информация (име, имейл, парола, адрес, телефон и специализация за ветеринарите).

- Търсене на ветеринари : Клиентите могат да филтрират ветеринари по име, град и специализация (домашни, екзотични или селскостопански животни).

- Резервиране на часове : Клиентите могат да видят наличните часове на ветеринарите (понеделник-петък, 9:00-17:00) и да резервират час.

- Подробна информация за ветеринари : Страница с детайли за всеки ветеринар, включваща снимка, адрес, рейтинг и график.

- Управление на данни : Базата данни съхранява информация за потребители, ветеринари, клиенти, градове, адреси, специализации, часове и рейтинги.

# Същинска част

# Глава 1: Проучвателна част

## 1. Преглед на съществуващи решения

В тази глава ще разгледаме някои от най-разпространените мобилни приложения за управление на здравето на домашни любимци, ще идентифицираме основните им силни и слаби страни и ще формулираме целите и задачите на нашия проект въз основа на направения анализ.

1.1 PetDesk

**Основни функционалности:**

Онлайн записване на часове и автоматични напомняния

Двупосочна комуникация между ветеринар и собственик

Достъп до медицински досиета

Проследяване на лекарства и ваксинации​

**Плюсове:**

Централизирана информация за домашния любимец

Подобрена комуникация с ветеринарния екип​

**Минуси:**

Ограничена персонализация на интерфейса

Някои функции изискват партньорство с конкретни клиники​

1.2 Vetstoria

**Основни функционалности:**

24/7 онлайн записване на часове

Персонализиране на наличностите и предпочитанията на лекарите

Интеграция с уебсайта на клиниката​

**Плюсове:**

Удобство за потребителите с възможност за записване по всяко време

Подобрена ефективност на клиниката чрез намаляване на телефонните обаждания​

**Минуси:**

Не предлага функции за управление на медицински досиета

Ограничена функционалност извън записването на часове​

1.3 VitusVet

**Основни функционалности:**

Достъп до медицински досиета

Автоматични напомняния за срещи и лекарства

Възможност за споделяне на информация с ветеринаря​

**Плюсове:**

Подобрена комуникация между собственик и ветеринар

Улеснен достъп до важна медицинска информация​

**Минуси:**

Ограничена функционалност за управление на рецепти

Липса на интеграция с други услуги като онлайн аптеки​

## 2. Анализ на проблемите и пропуските

### 2.1 Липса на цялостно решение

Повечето съществуващи приложения се фокусират върху една или две основни функции, като записване на часове или достъп до медицински досиета, но не предлагат цялостно решение, което обединява всички аспекти на грижата за домашния любимец.​

### 2.2 Ограничена персонализация

Много от приложенията не позволяват на потребителите да персонализират интерфейса или функционалността според специфичните нужди на техния домашен любимец.​

### 2.3 Липса на интеграция с други услуги

Ограничената интеграция с услуги като онлайн аптеки, застраховки или телемедицина ограничава полезността на приложенията за потребителите.​

# Подробен анализ на функционалностите на приложението VetTime

1. Записване на часове при ветеринар

• **Филтриране по критерии**: Потребителите могат да търсят ветеринари по **град**, **име** и **специализация** (домашни, екзотични, селскостопански животни).

• **График с налични часове**: Всеки ветеринар има дефиниран работен график (понеделник–петък, 09:00–17:00), в който клиентите могат да избират свободни часове.

• **Подробна страница за всеки ветеринар**: Включва снимка, адрес, специализация, рейтинг и достъпен график за резервация.

• **Динамично резервиране**: Клиентът избира слот, който автоматично се запазва и отбелязва като зает.

2. Регистрация и управление на потребители

• **Регистрация на клиенти и ветеринари**: Два различни потока за създаване на профили с подходящи полета.

• **ASP.NET Identity интеграция**: Сигурна система за автентикация с хеширани пароли.

• **Роли и права**: Разделение между клиенти и ветеринари за различен достъп до функционалности.

• **Автоматично създаване на свързани данни**: При регистрация се създават записи във Client или Veterinarian, обвързани с потребителя.

### 3. Управление на специализации и градове

• **Предварително зададени стойности**: Използват се seed-нати данни за **градове** и **специализации**, достъпни чрез падащи менюта.

• **Възможност за разширение**: Администратор (в бъдеща версия) може да добавя нови специализации или градове.

• **Ентитетни връзки**: Ветеринарите са свързани с една или повече специализации чрез таблицата VetSpecialization.

### 4. Администрация на данни и база

• **Entity Framework Core**: Използва се за работа с базата чрез Code First подход.

• **Seeding класове**: Автоматично създаване на тестови данни (градове, специализации, потребители и графици).

• **Композитни ключове и връзки**: Гарантира се уникалност

• **Soft delete логика**: Повечето модели поддържат маркиране на изтриване, без реално да се трие от базата.

5. Оценки и отзиви

• **Форма за оценка**: След посещение, клиентът може да оцени ветеринаря с оценка от 1 до 5.

• **Реализация с композитен ключ**: Един клиент може да даде само една оценка на един ветеринар.

• **Среден рейтинг**: Показва се автоматично в списъка с ветеринари и на страницата с детайли.

• **Прозрачност и доверие**: Отзивите помагат на нови потребители да направят информиран избор.

6. Визуално представяне и UX

• **Bootstrap дизайн**: Използване на карти за представяне на ветеринарите и responsive елементи. Използва се и за полета за регистрация.

• **Изчистен интерфейс**: Сортиране и филтриране без презареждане на цялата страница (чрез асинхронни POST заявка към Home/Recommendations).

• **Съобразено с потребителя**: Ясна навигация, контрастни бутони за действия и визуален календар.

# 

# Глава 2: Задание за софтуерна разработка

## 1. Функционални изисквания

### 1.1 Регистрация на ветеринари

Ветеринарите попълват специализирана форма с:

Име, имейл, телефон

Специализация

Адрес на клиника (град, улица, номер)

Работно време (понеделник – петък, 9:00 – 17:00)

Снимка и описание

### 1.2 Регистрация на клиенти

Клиентите предоставят:

Лични данни: име, имейл, телефон

Домашни любимци: име, вид, порода, снимка

### 1.3 Търсене и препоръки

Форма за търсене по град, специализация, име

Резултатите се визуализират в нова страница с карти на ветеринарите

Всяка карта включва:

Име

Адрес

Снимка на ветеринар (по желание)

Рейтинг

Бутон за детайли

### 1.4 Детайли за ветеринар

Подробна информация:

Работно време (график по дни и часове)

Възможност за резервиране на час

Отзиви от клиенти и оценка

### 1.5 Резервации

Клиент може да запази час за преглед

Ветеринарят вижда графика си по дни и часове

Записаните часове се блокират

### 1.6 Администраторски панел

Преглед на всички потребители, ветеринари и клиенти

Редакция/изтриване на профили

Справки за брой резервации, градове и специализации

## 2. Нефункционални изисквания

### 2.1 Сигурност

ASP.NET Identity

HTTPS защита

### 2.2 Производителност

Страници с резултати използват пагинация

Заявки, оптимизирани чрез LINQ + EF Core

### 2.3 Надеждност и поддръжка

Централизирана обработка на грешки

Страници при грешки 404 и 500

Unit тестове за Service слоеве

### 3. Избор на технологии

|  |  |
| --- | --- |
| **Слой** | **Технология** |
| Презентация | ASP.NET Core MVC + Razor Views |
| Автентикация | ASP.NET Identity |
| ORM / База данни | Entity Framework Core + MS SQL Server |
| UI библиотека | Bootstrap 5 |
|  |  |
| CI/CD | GitHub + Docker |
| DI | Вграден в ASP.NET Core |

## 4. Моделиране на системата

### 4.1 Основни Entities

ApplicationUser

Client

Veterinarian

Address

Animal

Appointment

Specialization

Rating

### 4.2 Връзки между основните модели

ApplicationUser ↔ Client (1:1)

ApplicationUser ↔ Veterinarian (1:1)

Client ↔ Animal (1:M)

Client ↔ Appointment (1:M)

Client ↔ Rating (1:M)

Veterinarian ↔ Rating (1:M)

* Veterinarian ↔ Appointment (1:M)

Veterinarian ↔ Specialization (M:M)

## 5. Представяне (Views)

### 5.1 Home

Index.cshtml (с форма за търсене)

Recommendations.cshtml (резултати от търсене)

### 5.2 Authentication

Login.cshtml

Register.cshtml (за клиенти)

VetRegister.cshtml (за ветеринари)

* Details.cshtml (с график, бутон за резервация и детайли за ветеринар)

## 6. ViewModels

* HomeViewModel
* VetViewModel
* VetDetailsViewModel
* AppointmentSlotViewModel
* SpecializationViewModel
* RegisterModel
* VetRegisterModel
* ErrorViewModel

7. Контролери

HomeController

VetController

# 

# Трета глава:

# Проектиране, имплементация, тестване и внедряване

## 3.1 Проектиране на системата

При проектирането на VetTime – уеб приложение за проследяване на ветеринарни посещения – беше приложен структуриран и многослоен подход. Проектът следва добри практики в софтуерното инженерство с акцент върху:

**Разделение на отговорностите (Separation of concerns):** Системата е структурирана в отделни слоеве – VetTime.Data (модели, конфигурации и база данни), VetTime.Services (бизнес логика), VetTime.Web (контролери, изгледи и CSS) и VetTime.Web.ViewModels (модели за изгледи).

**Разширяемост:** Всички основни операции като търсене на ветеринари, управление на графици, обработка на специализации и адреси са организирани чрез услуги (services). Това позволява лесно добавяне на нови функционалности, като например напомняния за предстоящи прегледи или администраторски модул.

**Тестваемост:** Услугите са дефинирани чрез интерфейси и се използва Dependency Injection. Това прави възможно лесно създаване на mock обекти и unit тестове, без нужда от реална база данни.

## 3.2 Архитектура на приложението (MVC)

VetTime използва архитектурния модел **Model-View-Controller (MVC)**, който разделя логиката на три основни компонента:

**Models:** Представят бизнес логиката и структурата на данните в базата. Включват класове като Veterinarian, Client, Appointment, Animal, Specialization, Address, City, VetSpecialization, Rating, Breed има и енумерация AppointnmentType за видове Appointment.

**Views:** Създадени с Razor (.cshtml), отговарят за визуализиране на информация към потребителя. Страниците са организирани в папки (Home, Vet, Shared и др.), включително частични изгледи и стилови файлове (site.css, schedule.css).

**Controllers:** Посредничат между изгледите и услугите. Основните контролери са:

HomeController – начална страница, търсене и препоръки

VetController – детайли за ветеринар и резервации

**Services:** Съдържат бизнес логика, например:

VetService – добавяне на ветеринар, филтриране, детайли

AddressService, CityService, SpecializationService – работа с адреси, градове, специализации

**Data:** Проектът съдържа ApplicationDbContext, който дефинира DbSet<T> за всички модели. Използва се Entity Framework Core с подход Code-First и миграции за управление на схемата.

## 3.3 Моделиране на данните (словесна ER диаграма)

## Основни таблици и връзки: (figure 1)

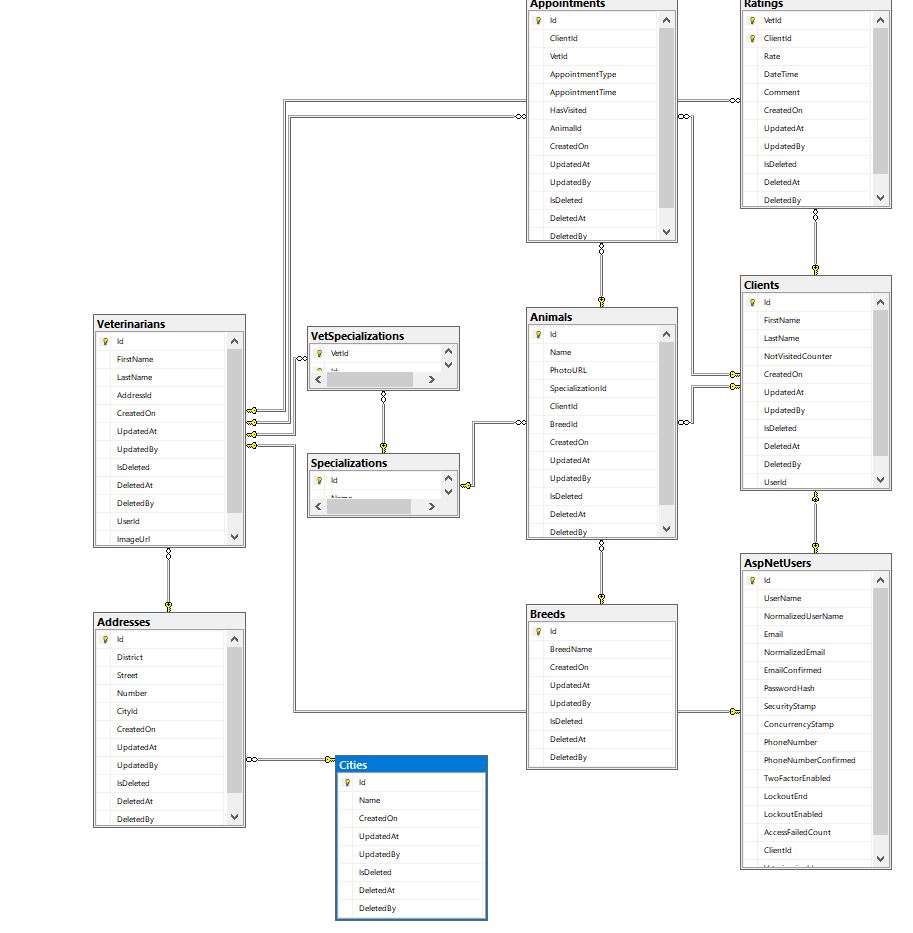


figure 1

ApplicationUser ↔ Client (1:1)

ApplicationUser ↔ Veterinarian (1:1)

Client ↔ Animal (1:M)

Client ↔ Appointment (1:M)

Client ↔ Rating (1:M)

Veterinarian ↔ Appointment (1:M)

Veterinarian ↔ Rating (1:M)

Veterinarian ↔ Specialization (M:M)

чрез VetSpecialization (mapping table)

City ↔ Address (1:M)

Address ↔ Veterinarian (1:M)

Specialization ↔ Animal (1:M)

## 3.4 Основни действия в приложението

## Регистрация и влизане

* Потребителят избира дали да се регистрира като клиент или ветеринар. (figure 2)

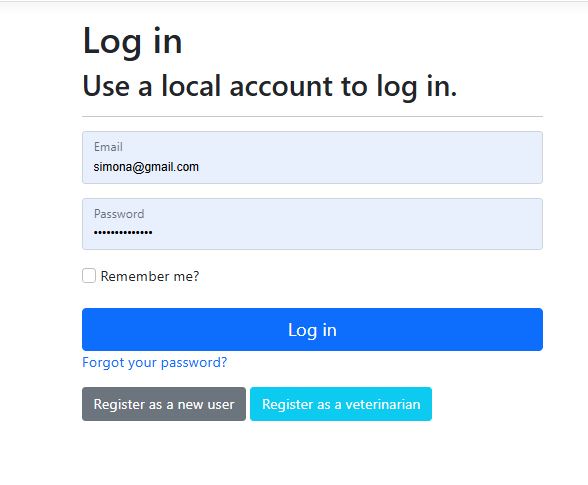


figure 2

При влизане със вече регистриран профил се попълват полета за имейл и парола. (figure 2)

При регистриране като ветеринарен лекар, тогава той се регистрира по първо и последно име, имейл, парола, телефонен номер, Адрес-град (dropdown), квартал, улица и номер, регистрира и каква специализация е (checkbox с три варианта - domestic animals, exotic

animals, farm animals като може да се регистрира с поне едно попълнено). (figure 3)

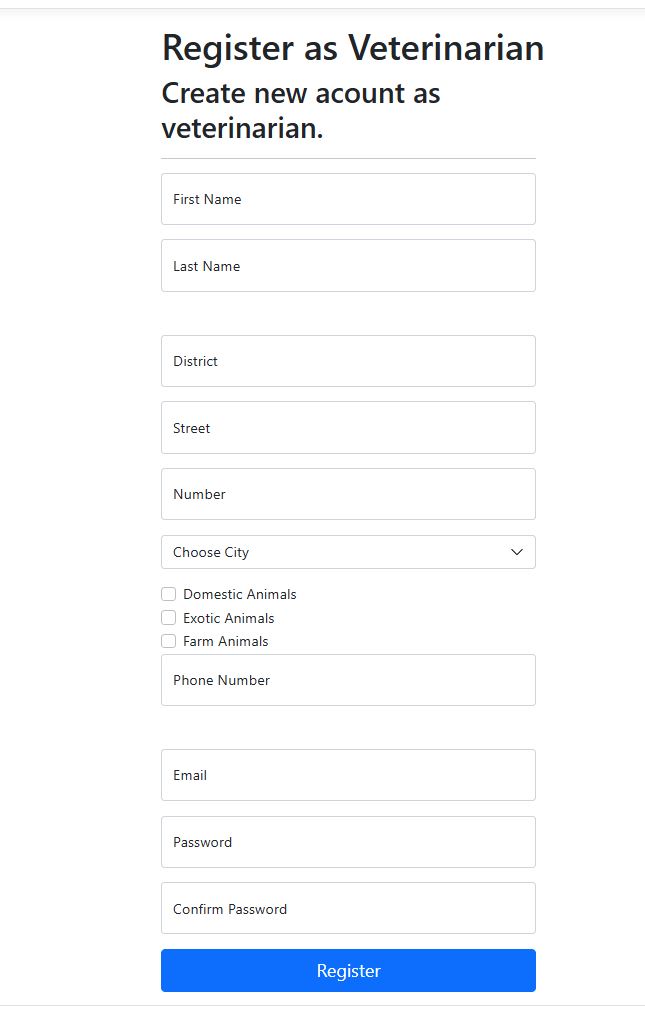


figure 3

При регистриране като клиент, той въвежда своето първо име, фамилия, имейл и парола. (figure 4)

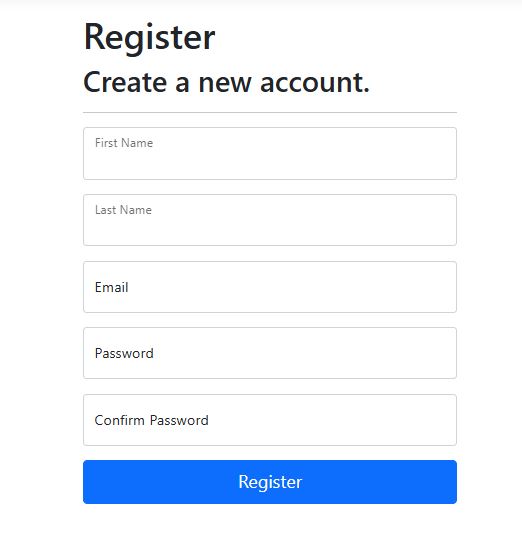


figure 4

## Търсене на ветеринари

* Началната страница съдържа форма за търсене по **първо и второ** **име**, **град** и **специализация**. Ако не е попълнен нито един от филтрите, при натискане на бутон ,,Search” ще се покажат всички регистрирани в приложението ветеринарни лекари. Но и ако е написана само една буква в полето за имена, ще се покажат всички ветеринари, съдържащи тази дума в първото или второто си име. Това се постига чрез използването на Wildcard в заявките за филтриране. (figure 5)

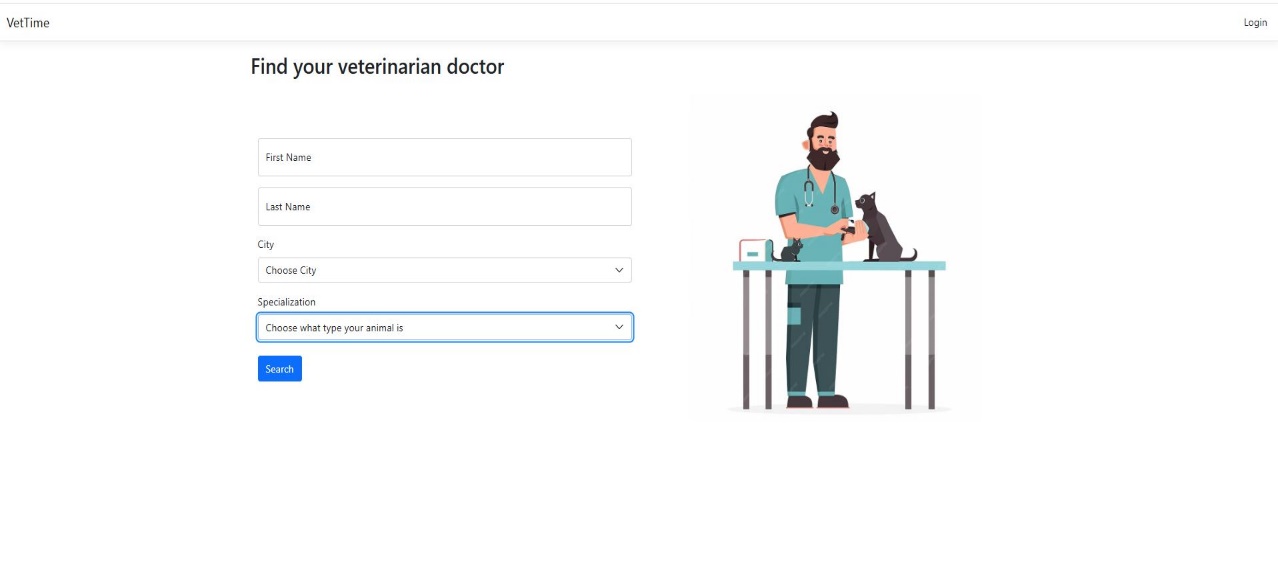


figure 5

* При изпращане на заявка (POST), системата пренасочва към страница

Recommendations, където се показват резултатите (figure 6). Където отново има полета за филтрация, в случай че клиента иска да промени или добави някоя филтрация.

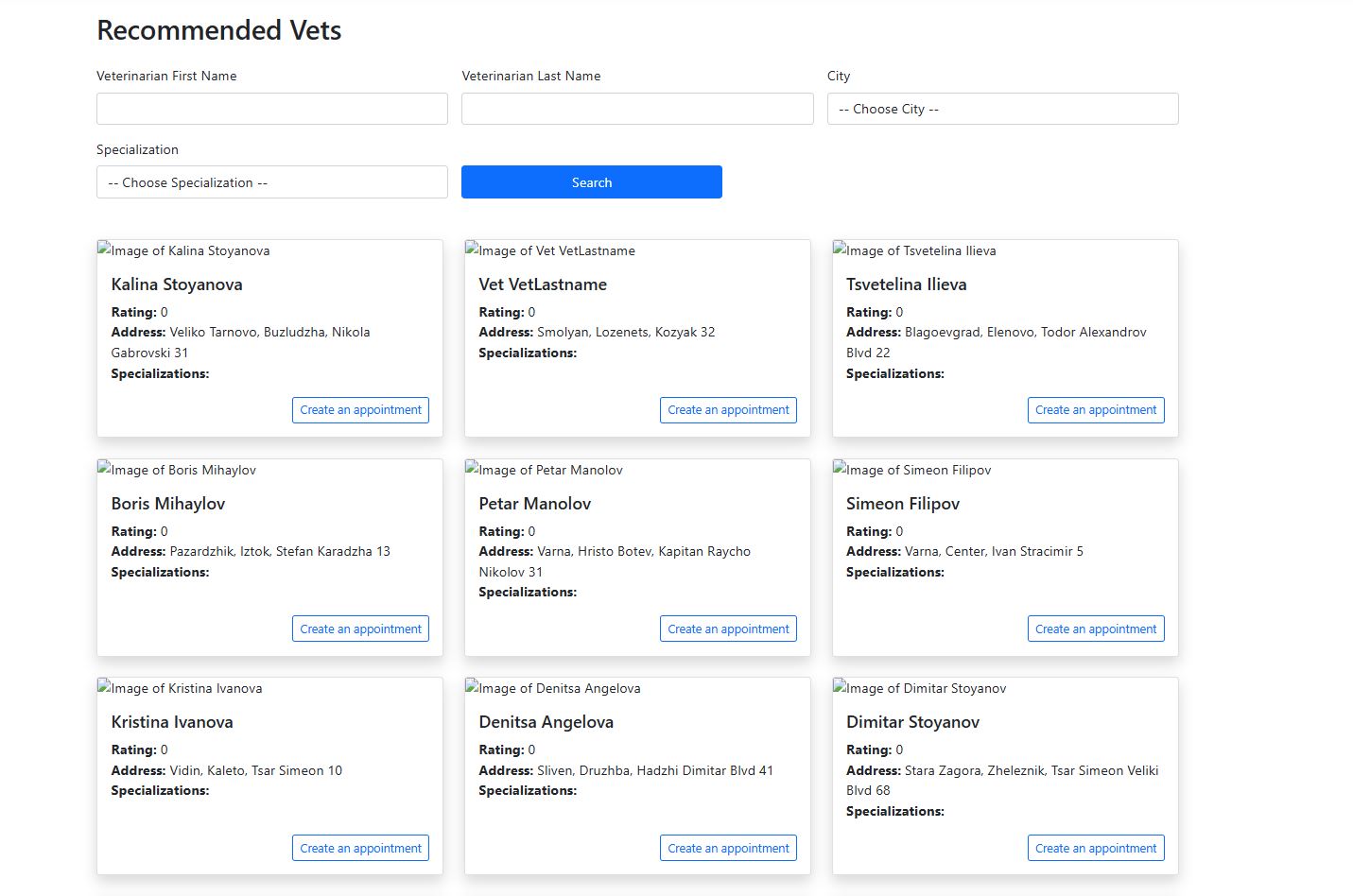


figure 6

* Функцията vetService.GetVetsInformation() връща филтриран списък.

## Преглед на детайли и резервация на час (figure 7)

В изгледа Details.cshtml се показват:

Име, снимка, адрес, специализации, рейтинг

График (понеделник – петък, 9:00–17:00 часа)

Всички часове са през един час разстояние.

Свободни и заети слотове, маркирани чрез CSS

Клиентът избира слот и изпраща заявка за резервация чрез BookAppointment.

Ако клиентът се опита да направи резервация за час, без да се е регистрира, при натискане на бутона ,,Save appointment” ще го препрати към страницата за регистриране. Ако резервацията е направена успешно, ще го върне към HomePage

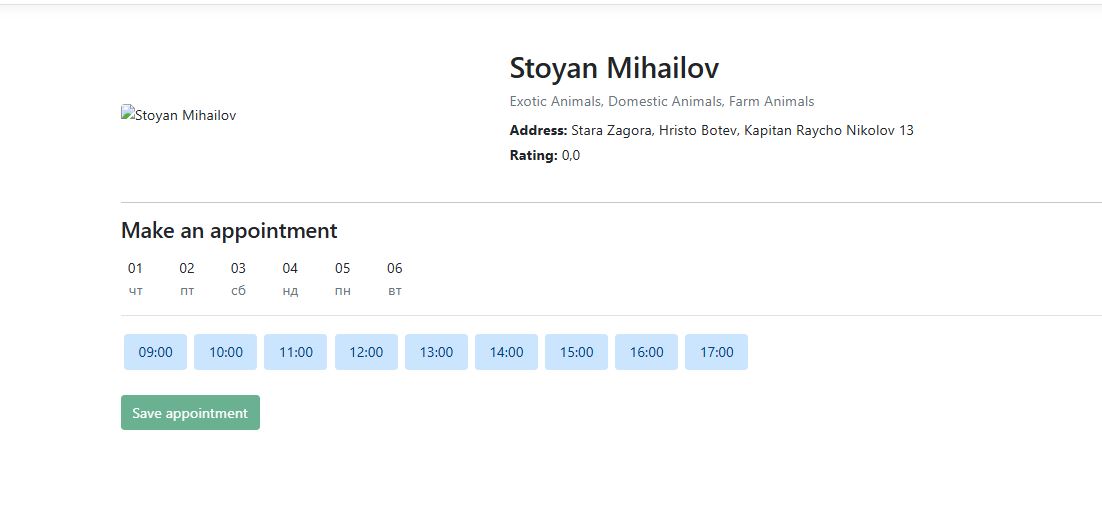


figure 7

## Добавяне и управление на ветеринарен профил

Ветеринарният профил се създава чрез VetRegisterModel, който се подава към VetService.Add().

## 3.5 Имплементация и използвани технологии

|  |  |
| --- | --- |
| **Слой** | **Технология** |
| Презентация | ASP.NET Core MVC + Razor Views |
| Автентикация | ASP.NET Core Identity |
| ORM | Entity Framework Core (Code-First) |
| База данни | Microsoft SQL Server |
| Интерфейс (UI) | Bootstrap 5 + CSS |
| Имейли (опция) | IEmailSender с SMTP |
| Хостване | Подготвен за IIS / Docker |

## 3.6 Тестване

* **Основни функционалности са валидирани:**
  + Регистрация и влизане
  + Търсене по филтри
  + Преглед и резервация на часове
* **Използване на Seeder-и:**

AddressSeeder

ApplicationUserSeeder

AppointmentSeeder

CitySeeder

ClientSeeder

SpecializationSeeder

VeterinarianSeeder

VetSpecializationSeeder

генерират тестови данни

* + Това позволява реалистично зареждане на базата и демонстрация

### **Тестове с реална база:**

* + Потвърдено е, че филтрацията работи правилно, че графикът се зарежда коректно и че връзките между моделите са функционални

## 3.7 Внедряване

### **База данни:**

* + Използват се EF Core миграции за създаване и поддръжка на схемата

### **Готовност за бъдещи надграждания:**

* + Примерни разширения включват:
    - Управление на профили от администратор
    - Прозорец за редактиране на профил

### Глава4: Архитектура и структура на проекта VetTime​

### VetTime е уеб приложение, изградено с ASP.NET Core MVC и Entity Framework Core, предназначено за управление на ветеринарни посещения. Архитектурата му следва принципите на разделение на отговорностите, модулност и лесна поддръжка. Проектът е структуриран в няколко основни слоя:​

## 4.1 VetTime.Data

### Този проект съдържа моделите на данни, конфигурациите и логиката за инициализация на базата данни.​

Configurations:

* Настройват връзките между моделите и базата данни, както и първоначалните данни чрез HasData.
  + Примери:
    - AddressConfiguration: Добавя предварително зададени адреси.
    - ApplicationUserConfiguration: Създава тестови потребители.
    - AppointmentConfiguration: Добавя тестови часове.

Seeding:

* Класовете в тази папка попълват базата данни с начални данни, полезни за тестване и демонстрация.
  + Примери:
    - AddressSeeder: Генерира примерен адрес.
    - ApplicationUserSeeder: Създава тестови потребители.
    - AppointmentSeeder: Генерира примерен час.

ApplicationDbContext:

* Основният контекст на базата данни, който управлява достъпа до данните и конфигурира връзките между моделите.​

## 4.2 VetTime.Data.Models

Съдържа класовете, представящи същностите в базата данни, както и един енумератор:​

### **Модели**:

Address: Представлява адрес с връзка към City.

Animal: Представлява животно с връзки към Client и Specialization.

ApplicationUser: Разширява IdentityUser с връзки към Client и Veterinarian.

Appointment: Представлява час с връзки към Client, Veterinarian и Animal.

Breed: Представлява порода на животно.

City: Представлява град с връзка към Address.

Client: Представлява клиент с връзки към ApplicationUser, Animal, Appointment и Rating.

Rating: Представлява оценка от клиент за ветеринар.

Specialization: Представлява специализация с връзки към Veterinarian и Animal.

Veterinarian: Представлява ветеринар с връзки към ApplicationUser, Address, Appointment, Rating и Specialization.

VetSpecialization: Свързва ветеринар със специализация.​

### **Enums**:

AppointmentType: Дефинира типове часове

## 4.3 VetTime.Services

Този проект съдържа интерфейси и услуги, управляващи бизнес логиката на приложението.​

### **Интерфейси**:

IAddressService: Метод Add за добавяне на адрес.

ICityService: Методи Add и GetAllCityNames за управление на градове.

ISpecializationService: Методи GetAllSpecializations и AddVetSpecialization за управление на специализации.

IVetService: Методи Add, GetVetsInformation и GetVetDetails за управление на ветеринари.​

### **Услуги**:

AddressService: Добавя адрес, ако не съществува.

CityService: Управлява градовете в базата данни.

SpecializationService: Управлява връзките между ветеринари и специализации.

VetService: Управлява ветеринарите и техните данни.​

## 4.4 VetTime.Web

Съдържа контролерите, изгледите и CSS файловете за потребителския интерфейс.​

### **Контролери**:

HomeController: Управлява началната страница и търсенето.

VetController: Управлява функционалностите за ветеринари.​

### **Изгледи**:

Login.cshtml: Страница за влизане.

Register.cshtml: Страница за регистрация на клиенти.

VetRegister.cshtml: Страница за регистрация на ветеринари.

Index.cshtml: Начална страница с форма за търсене.

Details.cshtml: Страница с детайли за ветеринар и график за резервации.

Recommendations.cshtml: Страница с резултати от търсене на ветеринари.​

### **CSS**:

site.css: Глобални стилове за шрифтове, бутони и менюта.

schedule.css: Стилове за графика на часовете.​

## 4.5 VetTime.Web.ViewModels

Съдържа модели за изгледи, улесняващи трансфера на данни между контролери и изгледи.​

### **Home**:

HomeViewModel: Модел за началната страница и търсенето.

SpecializationViewModel: Модел за специализация.​

### Vet:

AppointmentSlotViewModel: Модел за слот за час.

VetDetailsViewModel: Модел за детайли за ветеринар.

VetViewModel: Модел за ветеринар в списък.​

# Заключение

### Проектът **VetTime** демонстрира възможностите на съвременното уеб разработване чрез създаване на цялостна и функционална система за управление на ветеринарни посещения. Благодарение на архитектурата Model-View-Controller и използването на ASP.NET Core MVC, приложението е гъвкаво, структурирано и лесно за разширяване.

### VetTime позволява ефективна комуникация между ветеринарни лекари и техните клиенти, като предлага търсене по критерии, визуален график за резервации, профили на ветеринари и възможност за оценка. Интегрирането на Entity Framework Core (Code-First) и SQL Server улеснява управлението на данните, а внедряването на DI и интерфейси прави системата лесна за тестване и поддръжка.

### Чрез внедрените seed-данни приложението е готово за демонстрация и тестване в реална среда. То може лесно да бъде хоствано и поддържа бъдещи разширения, като администраторски контролен панел, имейл известия и допълнителни роли.

### Създаването на VetTime не само покрива изискванията за дипломна работа, но и демонстрира задълбочени познания по проектиране на софтуер, бази от данни, UI/UX и сигурност. Приложението има потенциал за реална употреба в клинична практика и предоставя надеждно и удобно решение за ветеринарни грижи в дигиталната ера.

# Източници

 [Entity Framework Core - Официална документация](https://learn.microsoft.com/en-us/ef/core/)

 [ASP.NET Core MVC с EF Core - Учебна серия](https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/?view=aspnetcore-9.0)

 [Entity Framework Core - Уроци и ръководства](https://www.learnentityframeworkcore.com/)

* [Introduction · Bootstrap](https://getbootstrap.com/docs/4.1/getting-started/introduction/)