**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

Факултет: Природни науки и образование

Катедра: Информатика и информационни технологии

**ДОПУСКАМ ДО ЗАЩИТА**

Ръководител на специали-

зиращата катедра:……..………

(доц.д-р Румен Русев)

**ОБЯСНИТЕЛНА ЗАПИСКА**

**НА ДИПЛОМЕН ПРОЕКТ**

**Тема**: **Инструмент за прогнозиране на крайна оценка по дисциплина.**

Дипломант: ………………… **ОДОБРЯВАМ**

(Божидар Атанасов) Ръководител:…………………… (Камелия Шойлекова)

Русе, 2025 г.

**Съдържание**

[1. Увод 5](#_Toc200582409)

[2. Обзор на съществуващи решения. Изводи. Цели и задачи 6](#_Toc200582410)

[**2.1. Разглеждане на съществуващи решения** 6](#_Toc200582411)

[**2.1.1. Cope** 6](#_Toc200582412)

[**2.1.2 Реколта digital** 7](#_Toc200582413)

[**2.1.3 Agrii** 8](#_Toc200582414)

[**2.1.4 Агро Калкулатор** 9](#_Toc200582415)

[**2.1.5 Preparati.info** 10](#_Toc200582416)

[**2.2 Анализ на съществуващите системи** 11](#_Toc200582417)

[**2.3 Изводи. Цели и задачи** 11](#_Toc200582418)

[**2.3.1 Изводи** 11](#_Toc200582419)

[**2.3.2 Цел и задачи** 11](#_Toc200582420)

[**3.1 Архитектурен модел на системата** 12](#_Toc200582421)

[**3.1.1 Използвани технологии** 12](#_Toc200582422)

[**3.1.1.4 React** 13](#_Toc200582423)

[**3.1.1.5 NextJS** 13](#_Toc200582424)

[**3.1.1.6 Prisma** 13](#_Toc200582425)

[**3.1.1.7 PostgreSQL** 13](#_Toc200582426)

[**3.2 Архитектура на системата** 16](#_Toc200582427)

[**3.2.1 Middleware Pipeline конфигурация** 16](#_Toc200582428)

[**3.2.2 Обработка на заявката** 16](#_Toc200582429)

[**3.2.3 Автентикация и оторизация** 16](#_Toc200582430)

[**3.2.4 Prisma интеграция** 16](#_Toc200582431)

[**3.2.5 Redux управление на състоянието** 16](#_Toc200582432)

[**3.2.6 Компонентна структура** 16](#_Toc200582433)

[**3.2.7 Сигурност** 16](#_Toc200582434)

[**3.2.8 UI и стилизация** 16](#_Toc200582435)

[**3.3 Логически модел на системата** 17](#_Toc200582436)

[**3.3.1 Диаграми на случаите на употреба** 17](#_Toc200582437)

[**3.3.2 Диаграми на дейностите – Activity Diagrams** 19](#_Toc200582438)

[4. Описание на базата данни 20](#_Toc200582439)

[**4.1 Релационен модел на базата данни** 20](#_Toc200582440)

[**4.2. Описание на таблиците и връзките между тях** 21](#_Toc200582441)

[**4.2.1 Таблица Users** 21](#_Toc200582442)

[**4.2.2 Таблица UserSettings** 21](#_Toc200582443)

[**4.2.3 Таблица RefreshToken** 22](#_Toc200582444)

[**4.2.4 Таблица ResetPassword** 22](#_Toc200582445)

[**4.2.5 Таблица Plant** 22](#_Toc200582446)

[**4.2.6 Таблица SowingRatePlant** 22](#_Toc200582447)

[**4.2.7 Таблица SowingRateHistory** 23](#_Toc200582448)

[**4.2.8 Таблица SeedingDataCombination** 23](#_Toc200582449)

[**4.2.9 Таблица SeedingDataCombinationHistory** 23](#_Toc200582450)

[**4.2.10 Таблица Chemical** 23](#_Toc200582451)

[4.2.11 Таблица ChemicalActiveIngredient 23](#_Toc200582452)

[**4.2.12 Таблица ActiveIngredient** 24](#_Toc200582453)

[**4.2.13 Таблица Enemy** 24](#_Toc200582454)

[**4.2.14 Таблица ChemicalToEnemy** 24](#_Toc200582455)

[**4.2.15 Таблица PlantChemical** 24](#_Toc200582456)

[**4.2.16 Таблица ChemProtWorkingSolutionHistory** 24](#_Toc200582457)

[**4.2.17 Таблица ChemProtPercentHistory** 24](#_Toc200582458)

[5. Програмна реализация 25](#_Toc200582459)

[**5.1 Инжектиране на зависимости в Razor Pages (Dependency Injection)** 28](#_Toc200582460)

[**5.2 Основни страници** 28](#_Toc200582461)

[**5.2.1 Начална страница (Index)** 28](#_Toc200582462)

[**5.2.2 Страница Register** 30](#_Toc200582463)

[**5.2.3 Страница Logout** 31](#_Toc200582464)

[**5.2.4 Страница Admin – BrowseGrades** 32](#_Toc200582465)

[**5.2.5 Страница Admin - BrowseSemesters** 34](#_Toc200582466)

[**5.2.6 Страница Admin – BrowseStudents** 35](#_Toc200582467)

[**5.2.7 Страница Admin - BrowseSubjects** 35](#_Toc200582468)

[**5.7.8 Страница Admin - BrowseTeachers** 36](#_Toc200582469)

[**5.2.9 Страница Admin - Index** 36](#_Toc200582470)

[**5.2.10 Страница Student - SemesterOverview** 37](#_Toc200582471)

[**5.2.11 Страница Student - Index** 39](#_Toc200582472)

[**5.2.12 Страница Teacher – BrowseSubjects** 40](#_Toc200582473)

[**5.2.13 Страница Teacher – CourseDetails** 41](#_Toc200582474)

[**5.2.14 Страница Teacher – Index** 41](#_Toc200582475)

[**5.2.15 Страница Shared** 42](#_Toc200582476)

[6. Ръководство за работа със системата 42](#_Toc200582477)

[**6.1 Инсталиране на системата.** 42](#_Toc200582478)

[**6.1.1 Нужни ресурси за инсталирането на системата** 42](#_Toc200582479)

[**6.1.2 Стартиране на проекта** 43](#_Toc200582480)

[**6.2 Регистрация на нови потребители** 44](#_Toc200582481)

[**6.3 Употреба на приложението като студент** 45](#_Toc200582482)

[**6.4 Употреба на приложениетж като преподавател** 46](#_Toc200582483)

[**6.5 Употреба на приложението като администратор** 50](#_Toc200582484)

[7. Основни резултати, изводи и препоръки 55](#_Toc200582485)

[8. Използвана литература 55](#_Toc200582486)

[9. Таблица с фигури 57](#_Toc200582487)

[10. Таблица с таблици 58](#_Toc200582488)

[[2] AdminPlus, URL: https://adminplus.bg/, посетен на 07.02.2024г. 58](#_Toc200582489)

[[3] Administrator's Plus, URL: https://www.rediker.com/, посетен на 07.02.2024г. 58](#_Toc200582490)

[[4] ASP.NET Core, URL https://dotnet.microsoft.com/en-us/apps/aspnet, посетен на 10.02.2024г. 58](#_Toc200582491)

[[5] Microsoft SQL Server, URL: https://www.microsoft.com/en-us/sql-server/sql-server-downloads, посетен на 10.06.2024г. 58](#_Toc200582492)

[[6] Entity Framework Documentation Hub, URL: https://learn.microsoft.com/en-us/ef, посетен на 10.06.2024г. 58](#_Toc200582493)

[[7] Какво представлява унифицирания език за моделиране (UML), URL: https://www.microsoft.com/bg-bg/microsoft-365/visio/uml, посетен на 20.07.2024г. 58](#_Toc200582494)

# **1. Увод**

Фермерство, дори в тази модерна ера на квантови компютри, изкуствен интелект и телефони по-мощни от настолни компютри, един човек не може да живее без храна. В миналото всеки е разполагал с площ за фермерство или е плащал на друг за да може да се изхранва. С това е изхранвал както себе си така и семейството си. В днешно време това също е възможно, но повечето хора извършват тази дейност не, за да се изхрани, а като хоби през уикенда.

В днешно време фермери с хиляди декари площ, притежавана от тях или плащайки рента на нея, извършват тази дейност. От това зависи тяхното преживяване, инвестиции и дори в някой случай преживяването на други хора. Те не могат да правят грешки и трябват да намалят случайните фактори до миниум. За това повечето се консултират с агрономи и използват калкулатори [1].

Този проект показва, че използване на изкуствения интелект няма да замени агрономите. Идеята и темата за калкулиране на параметри е много стара. Едни от основните неща, които един фермер трябва лесно да може да калкулира са: сеитбената норма (колко растения трябва да посади от семки, за да поникнат толкова колкото очаква), сеитбена норма за смески, с какви препарати, с какви дози и как да пръска при борба с неприятели. Също да е запознат с одобрени за използване на Българския пазар препарати от ББХ (Българска агенция по безопасност на храните) (<https://bfsa.egov.bg/wps/portal/bfsa-web/home>).

Системата ще поддържа три вида калкулатора:

* калкулатор за сеитбената норма за различни култури - грах, соя, сорго, царевица, люцерна, звездан, червена детелина, бяла детелина, пасищен райграс, гребенчат житняк, ежова главица, безосилеста овсига, ливадна власатка и червена власатка ;
* калкулатор за сеитбената норма за смески;
* калкулатор за препарати с борба с неприятели;

Също ще поддържа:

* подробна информация за предлаганите от системата препарати;
* система за профили, в която ще се запазва информацията с резултатите от пресмятанията на конкретния потребител.

# **2. Обзор на съществуващи решения. Изводи. Цели и задачи**

## **2.1. Разглеждане на съществуващи решения**

### **2.1.1. Cope**

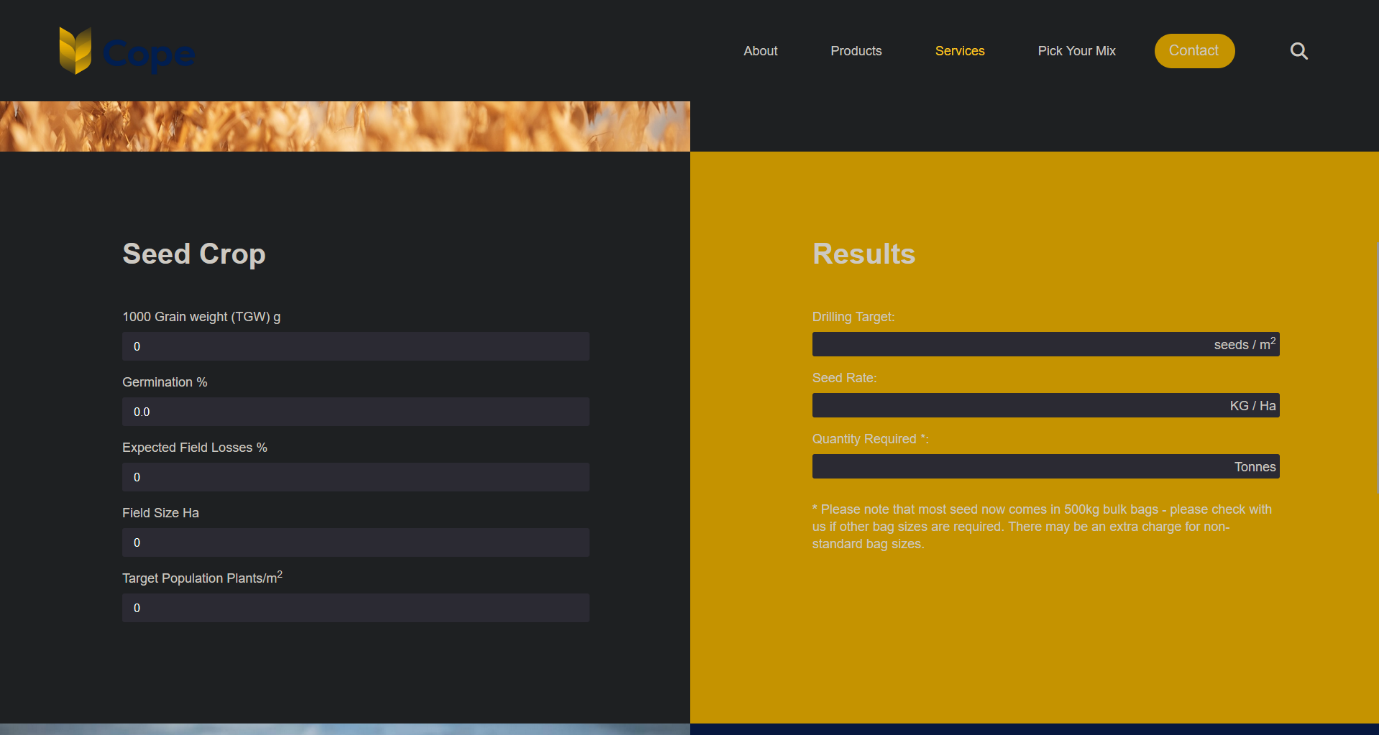
Страница направена от английска фирма занимаваща се с продажбата на семена и торове (Фигура 1) [2]. Сайтът е главно насочен към продажби, като допълнение има базов калкулатор, който не разполага с научно доказани данни. Продукта може да се види на.

**Предимства:**

* Малко възможности води до лесна употреба.
* На английски език е и е достъпен до друг пазар извън България.

**Недостатъци**:

* Липсват данни от резултата.
* Използва хектари, единица главно използвана в Англия.



Фигура Cope

### **2.1.2 Реколта digital**

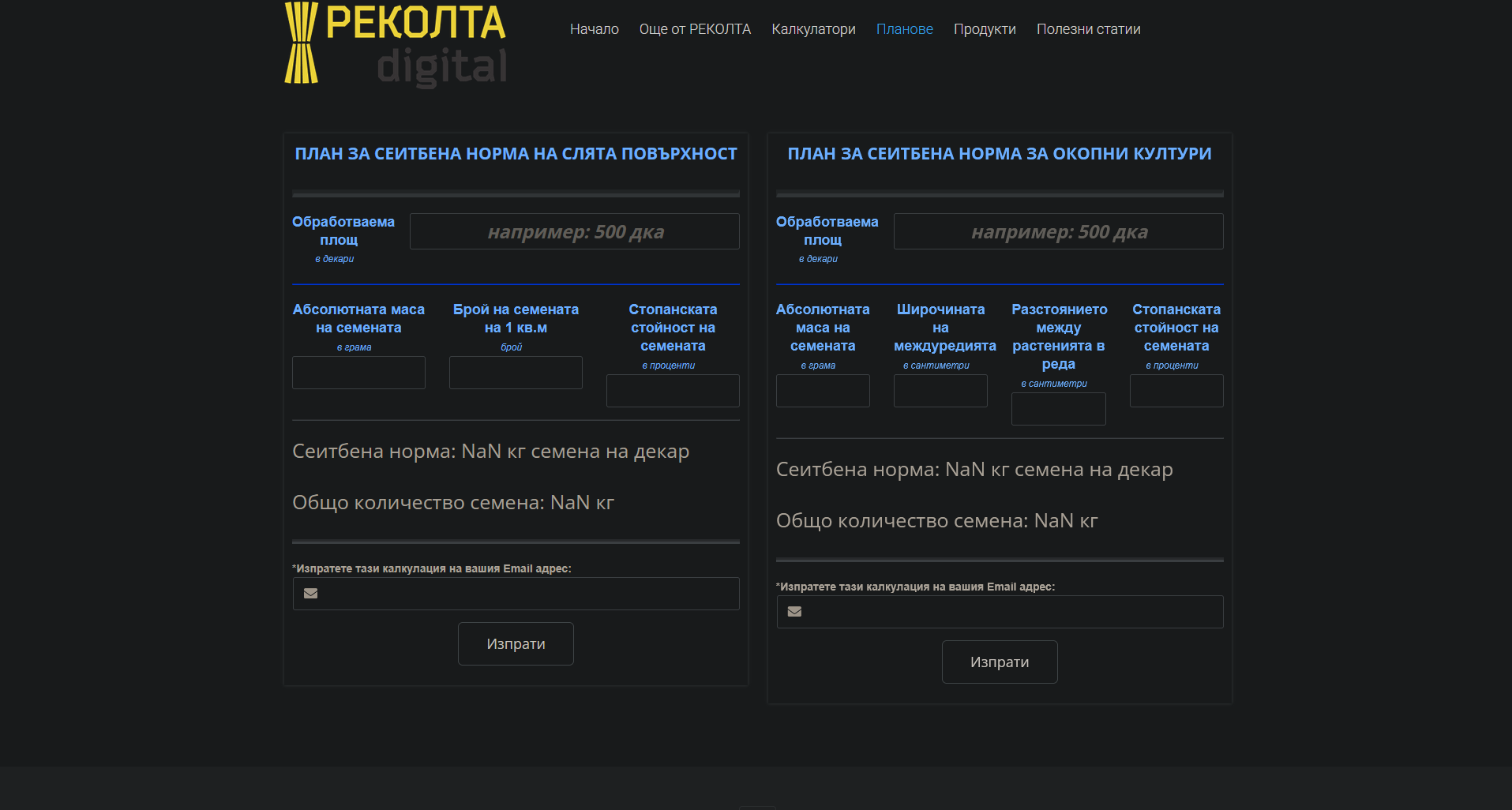
Подобно на Cope, Реколта digital също предлага подобен калкулатор за изчисляване на сеитбена норма за основни култури като пшеница и рапица засягащи всякакви черти от отглеждането на тези култури [3].

Предимства:

* на български език, като изчисленията са представени за улеснение в български единици;
* допълнителна информация и калкулатори за основни култури;
* може да бъде запазен резултата чрез изпращане по имейл;

Недостатъци:

* калкулаторът калкулира малко информация за сеитбени норми;
* няма информация за сеитбена норма за смески или за пръскане на различните култури с конкретен препарат за растителна защита.



Фигура . Реколта digital

### **2.1.3 Agrii**

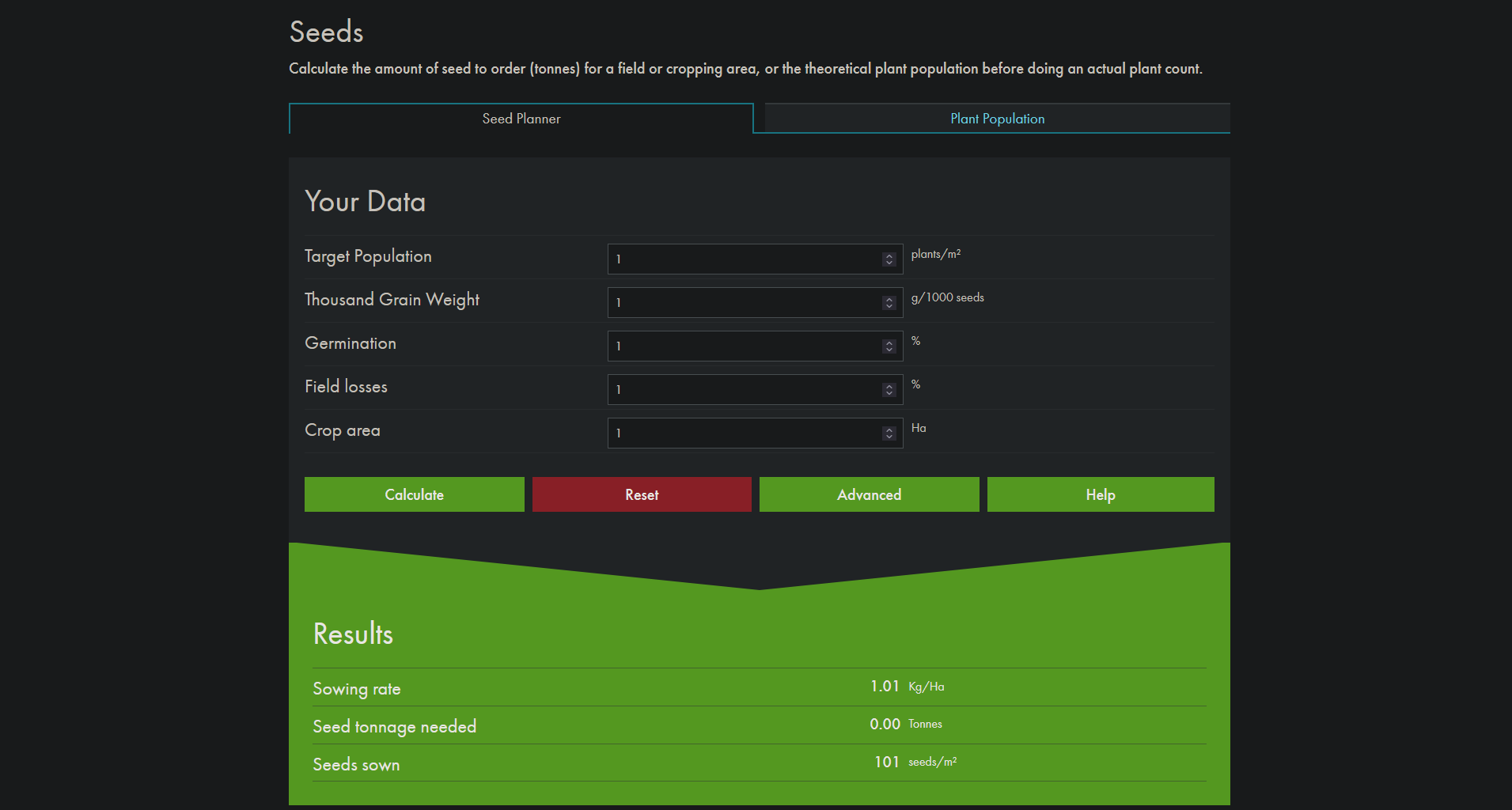
Agrii, подобна на Cope, е английска фирма занимаваща се с консултация и продажба на семена и торове. Основната цел на сайтът е продажба, а калкулаторът е базов като при Cope [4].

**Предимства:**

* добре направен с дизайн и функционалност;
* има помощно меню с обяснение на всяко поле какво прави;
* има възможност за запазване на изчисленията в имейл или да бъдат изтеглени като PDF;

**Недостатъци:**

* на английски език е;
* работи в хектари;
* има малки отличителни грешки в UI/UX;



Фигура Agrii

### **2.1.4 Агро Калкулатор**

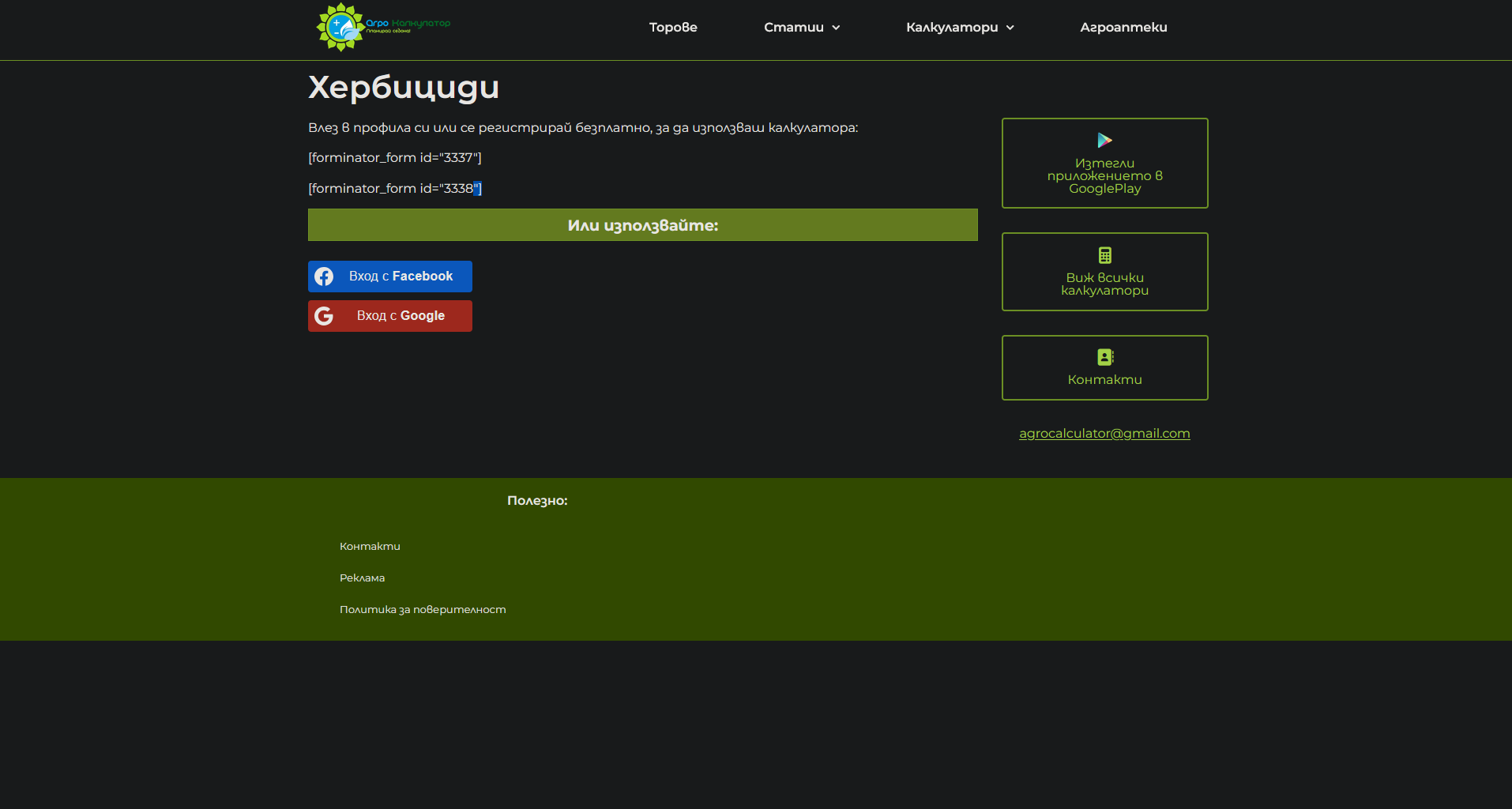
Agrocalculator.bg е български сайт, съвкупност от няколко калкулатора със земеделски цели. За съжаление не работи [5].

**Предимства:**

* има мобилно приложение.

**Недостатъци:**

* сайтът и калкулаторите не работят.



Фигура agrocalculator.bg

### **2.1.5 Preparati.info**

Preparati.info български сайт, съдържащ информация за препарати за растителна защита (ПРЗ) одобрени от ББХ да се използват в България [6].

**Предимства:**

* има много богата уикипедия с текущо използвани и забранени ПРЗ;
* има калкулатор за изчисляване на работен разтвор - една лесна, но всеки му нужна сметка;
* в калкулатора обяснява всяко поле за какво е;
* добър UI/UX;

**Недостатъци:**

* няма възможност за изчисляване на сеитбена норма при различни културни растения;
* няма възможност за запазване на направените изчисления.



Фигура preparati.info

## **2.2 Анализ на съществуващите системи**

Сравнение на функционалните възможности на различните продукти в таблица 1.

Таблица Сравнение на функционалности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Характеристики** | **Cope** | **Реколта digital** | **Agrii** | **agrocalculator.bg** | **preparati.info** |
| Уеб Базирана система | ДА | ДА | ДА | ДА | ДА |
| Предлагат повече от един калкулатор | НЕ | ДА | НЕ | ДА | НЕ |
| Предлага пресмятане на работен разтвор | НЕ | НЕ | НЕ | НЕ | ДА |
| Има достъп до научно потвърдени данни | НЕ | НЕ | НЕ | НЕ | ДА |

## **2.3 Изводи. Цели и задачи**

### **2.3.1 Изводи**

От направеният преглед на съществуващите решения се виждат някои недостатъци:

* не всички продукти имат повече от един калкулатор;
* не всички продукти предлагат комбинирани калкулатори позволяващи пресмятане на различни манипулации през периода на вегетация;
* не всички продукти включват фуражни култури;
* не всички продукти имат достъп до научно доказани данни.

### **2.3.2 Цел и задачи**

Целта на дипломния проект е да се разработи уеб базирана система за пресмятане на най-важните неща за земеделски стопанин и/или любител фермер, с достъп до научно доказани данни.

Разработваното на дипломния проект е практическо улеснение в помощ на широк кръг потребители (агрономи, земеделски производители, селскостопански специалисти, фермери, научни работници, студенти, докторанти, заинтересовани лица и др.) заложено в Проект ПОЗМ 12 „Оптимизиране на подхода за контрол на заплевеляването при основни земеделски култури“ с ръководител доц. Д-р Пламен Серафимов към Институт по декоративни и лечебни растения към Селскостопанска академия – София. Ще бъдат използвани обобщени научни изследвания на авторите, както и общодостъпни и общоизвестни правила, свързани с безопасното прилагане на препарати за растителна защита (ПРЗ) и изчисляване на сеитбени норми.

За постигането на тази цел са формулирани следните задачи:

* създаване на уеб базирано приложение, което да притежава лесен и достъпен дизайн за употреба;
* регистрация на потребители, управление на правата им на достъп;
* запазване на направени вече калкулации за по-късен анализ и употреба;
* показване на статистика за текущо направени и предишни калкулации.

**3. Архитектурен и логически модел на системата**

## **3.1 Архитектурен модел на системата**

### **3.1.1 Използвани технологии**

За създаването на дипломният проект са използвани следните технологии: Visual Studio Code, git, Docker, yarn, React, NextJS, Prisma, PostgreSQL, Typescript, react-redux, zod, Jest, ShadCN, Tailwind CSS, Lucide icons, Github, Github Actions, Netcup VPS и Cloudflare.

#### **3.1.1.1 Visual Studio Code**

Когато става въпрос за разработка, едно от най-важните неща са инструментите, които се използват. Visual Studio Code е гъвкава среда за разработка. То е текстов редактор с вградени инструменти и клавишни комбинации за лесна употреба от един програмист. В сравнение с пакети като Visual Studio и Eclipse даже има малко функционалности, но разполага с бурен и жив пазар за разширения (extensions), които го правят изключителна мощна среда позволяваща на един програмист да следи промените от написването им, до пускането им в производство (production) [7].

#### **3.1.1.2 git**

Проблем, който всеки програмист среща е запазване на версии на кода. В стари дни (и днес) файловете са на един споделен диск в облачно пространство или се пренасят от компютър на компютър с добавено допълнение към името \_final. Linus Torvalds предоставя решение на този проблем през 2005 година с FOSS (free and open source), безплатен с отворен код инструмент наречен git. Този инструмент позволява създаването на лесно разбираема нишка от промени, чрез който програмист може да работи спокойно, защото винаги има старата версия на проекта [8].

#### **3.1.1.3 Docker**

Docker е инструмент и платформа за контейнеризация, която позволява разработчиците да създават, разпространяват и стартират приложения в изолирани контейнери. Контейнерите осигуряват лека, преносима и последователна среда, която включва всички зависимости на приложението, което улеснява разработката, тестовете и внедряването в различни среди. Docker използва образи (images) за дефиниране на приложения и техните зависимости, а контейнерите се изпълняват от тези образи. Това елиминира проблеми, свързани с несъвместимости между операционни системи и библиотеки, подобрявайки мащабируемостта и ефективността на софтуерните решения [9].

## **3.1.1.4 React**

React е популярна JavaScript библиотека, разработена от Facebook, сега Meta, през 2013 година, която се използва за изграждане на динамични и интерактивни потребителски интерфейси. Тя следва компонентно-базиран подход, който улеснява повторното използване на код и поддръжката на големи приложения. Основно предимство на React е използването на виртуален DOM, който оптимизира обновяването на потребителския интерфейс, подобрявайки производителността. Освен това React поддържа еднопосочен поток на данни, което прави управлението на състоянието по-предвидимо. Благодарение на своята гъвкавост и мащабируемост, React е предпочитан избор за разработка на уеб и мобилни приложения [10].

## **3.1.1.5 NextJS**

Next.js е мощен JavaScript фреймуърк, базиран на React, разработен от Vercel, който улеснява създаването на бързи и оптимизирани уеб приложения. Той поддържа сървърно рендериране (SSR) и статична генерация (SSG), което подобрява производителността и SEO оптимизацията на сайтовете. Освен това, Next.js предлага автоматична маршрутизация, вградена поддръжка за API маршрути, лесна интеграция с CSS, Sass и Tailwind CSS, както и функции като динамични маршрути и инкрементално статично обновяване (ISR). Тези характеристики го правят отличен избор за изграждане на мащабируеми, интерактивни и кросплатформени уеб приложения [11].

## **3.1.1.6 Prisma**

Prisma е модерен ORM (Object-Relational Mapping) инструмент за бази данни, който улеснява работата с SQL бази като PostgreSQL, MySQL, SQLite, SQL Server и CockroachDB. Той предоставя удобен типизиран API за взаимодействие с базата данни, като автоматично генерира TypeScript или JavaScript код на базата на дефинирана схема. Prisma включва мощни инструменти като миграции на базата (Prisma Migrate), визуален интерфейс за заявки (Prisma Studio) и поддръжка за GraphQL и REST API. Благодарение на строгата типизация и интеграцията с TypeScript, Prisma подобрява сигурността и предвидимостта при работа с бази данни [12].

## **3.1.1.7 PostgreSQL**

PostgreSQL е гъвкава и мощна релационна база данни с отворен код, която работи на множество платформи, включително Linux, Windows и macOS. Тя е изключително стабилна, поддържа ACID транзакции, мащабиране и разширяемост чрез разширения. PostgreSQL е предпочитан избор за уеб, корпоративни и облачни приложения поради своята надеждност и висока производителност [13].

#### **3.1.1.8 TypeScript**

TypeScript е обектно-ориентиран език за програмиране, разработен от Microsoft и пуснат през 2012 година като надграждане на JavaScript. Основната му цел е да осигури по-добра структура и мащабируемост при разработката на уеб приложения, като добавя статична типизация и модерни функционалности. TypeScript е строго типизиран, което означава, че всяка променлива има конкретен тип, известен по време на компилацията. Това позволява на програмистите да откриват грешки още преди изпълнението (run time) на кода и улеснява поддръжката на големи проекти. За разлика от него, JavaScript е динамично типизиран език, което позволява по-гъвкаво, но и по-рисковано управление на променливите. Например, една променлива в JavaScript може да съдържа число (int), а по-късно да бъде заменена със низ (string), което може да доведе до грешки по време на изпълнение. TypeScript елиминира този проблем, като осигурява по-голяма предвидимост и стабилност при разработката на сложни приложения [14].

#### **3.1.1.9 react-redux**

react-redux е систем за пазене на state (състояние) в React. Поради факта че React работи от горе на долу за подаване на данни, например страница подава данни на елемент показващ информация за потребителя и изключително трудно за поддържане в обратна посока. Например на сайт за пазаруване където при натискане на бутон за добавяне в кошницата елемент в главата на сайта трябва да се обнови. react-redux предоставя „магазин“, в които чрез строго определени начин може да бъде запазена строго определена и форматирана информация, позволяваща на програмист да достъпва лесно и само с една команда да променя състоянието по определен начин [15].

#### **3.1.1.10 zod**

Zod е инструмент за валидиране на данни в Typescript чрез схема. Използва се главно за форми, за да даде сигурност на програмиста, че данните, които получава от уеб формата са в правилния формат и данните са валидирани на основата на създадени правила. При провал zod показва точно къде и как данните провалят своята валидация [16].

#### **3.1.1.11 Jest**

Jest е инструмент за извършване на тестове върху Javascript и Typescript код. Поддържа фреймуъркове като NextJS, React, Angular, Vue и други. Използва се от големи компании като Facebook, Spotify и Instagram. Дава възможност на програмистите да проверят по лесен начин, дали всяка функция и ред код извършват това, което очакват без ограничение в използваните параметри. Позволява симулация на действия, за да бъде тестван секция код без останалата част от програмата [17].

#### **3.1.1.12 ShadCN**

ShadCN e платформа с отворен код, с визуални компоненти за уеб страници, които работят заедно и имат документация. Изключително полезни за правенето на уеб приложение бързо и визуално изглеждащо добре. Вече притежавани от Vercel [18].

#### **3.1.1.13 Tailwind CSS**

Tailwind CSS е библиотека за стилизиране на уеб html елементи. Вместо разработчика да пише ръчно всеки стил в CSS, той може да използва и комбинира вече създадените стилове, за да персонализира своето приложение [19].

#### **3.1.1.14 Lucide icons**

Lucide е модерен, отворен код библиотека с икони, създадена с фокус върху лекотата, последователността и гъвкавостта. Тя предлага над 1500 висококачествени SVG икони, които са оптимизирани за използване в уеб и мобилни приложения. Проектът се поддържа от активна общност и е достъпен чрез различни пакети за популярни JavaScript рамки като React, Vue, Svelte, Angular и други [20].

#### **3.1.1.15 Github**

Github е частна платформа за инструмента git. Тя позволява лесно съхранение на кодови версии и споделяне на код. Самата платформа Github предоставя контрол на достъпа, система за следене на бъгове (грешки в програмата), запитвания за функционалност на програми, разпределяне на задачи между разработчици, система за автоматично строене на най-новата версия на кода и уикипедия със знания за всеки проект [21] [22].

#### **3.1.1.16 Github Actions**

Github Actions е в секцията за Github спомената система за автоматично строене на най-новата версия на кода. Може да се използва за строене на код, създаване на файлове и изпълнение на всяко действие, което може да бъде извършено от обикновен терминал. Представлява поредица от код който може да бъде визуално наблюдаван в Github извършващ задача [23].

#### **3.1.1.11 Netcup VPS**

Netcup VPS е услуга предлагана от фирмата netcup GmbH със седалище в Карлсруе, Германия. VPS (Virtual Private Server – Виртуален Частен Сървър) e услуга предлагаща облачно пространство, под формата на сървър работещ на Linux операционна система, напълно достъпно за клиент. Чрез тази услуга уеб сайта работи постоянно и може да бъде достъпен по всяко време на денонощието [24].

#### **3.1.1.11 Cloudflare**

Cloudflare е платформа предоставяща услуги за уеб сайтове като поддръжка на домейн, сигурност на сайта срещу атаки и лесно създаване на връзка между вървящия сайт на сървър и домейн [25].

## **3.2 Архитектура на системата**

Първата стъпка при Next.js е обработката на HTTP Request на изпратената заявка. Когато получим заявка към сървъра (например /profile), тя преминава през middleware pipeline (дефиниран в middleware.ts файла):

## **3.2.1 Middleware Pipeline конфигурация**

* Първо се дефинират защитените маршрути (protected routes) в routeDefinitions:
* API маршрути (/api/user/settings, /api/calc/\* и др.)
* Страници (/profile)
* Маршрути след автентикация (/auth/login, /auth/register)

## **3.2.2 Обработка на заявката**

* Проверява се дали заявката е към root path ('/')
* Определя се типът на заявката (API или страница)
* Проверява се дали маршрутът е защитен
* Извлича се access token от cookies

**3.2.3 Автентикация и оторизация:**

* При достъп до защитен маршрут се проверява наличието на access token
* Ако няма token:
* За API заявки връща 401 Unauthorized
* За страници пренасочва към /auth/login
* Ако има token:
* Валидира се чрез BackendVerifyToken
* При невалиден token се опитва refresh чрез BackendRefreshAccessToken
* При неуспешен refresh се извършва logout

## **3.2.4 Prisma интеграция**

* Използва се Prisma като ORM за връзка с базата данни
* Моделите са дефинирани в prisma/schema.prisma
* Достъпът до базата данни е централизиран чрез Prisma Client

## **3.2.5 Redux управление на състоянието**

* Глобалното състояние се управлява чрез Redux Toolkit
* Persist осигурява запазването на данните в браузъра
* Действията (actions) и редюсерите (reducers) са организирани в /store директорията

## **3.2.6 Компонентна структура**

* Използва се Next.js App Router
* Компонентите са организирани в /components и /app директории
* Поддържа Server Components за оптимизирано рендериране

## **3.2.7 Сигурност**

* JWT базирана автентикация
* Refresh token механизъм
* Защитени маршрути чрез middleware
* Хеширане на пароли и сигурно съхранение на данни

## **3.2.8 UI и стилизация**

* TailwindCSS за стилизация
* Radix UI компоненти за достъпен интерфейс
* Темизация чрез next-themes

Този архитектурен подход осигурява:

* Висока сигурност чрез middleware и JWT автентикация
* Добра производителност чрез Server Components
* Типова безопасност чрез TypeScript
* Лесна поддръжка чрез модулна структура
* Скалируемост чрез добре организиран код

## **3.3 Логически модел на системата**

### **3.3.1 Диаграми на случаите на употреба**

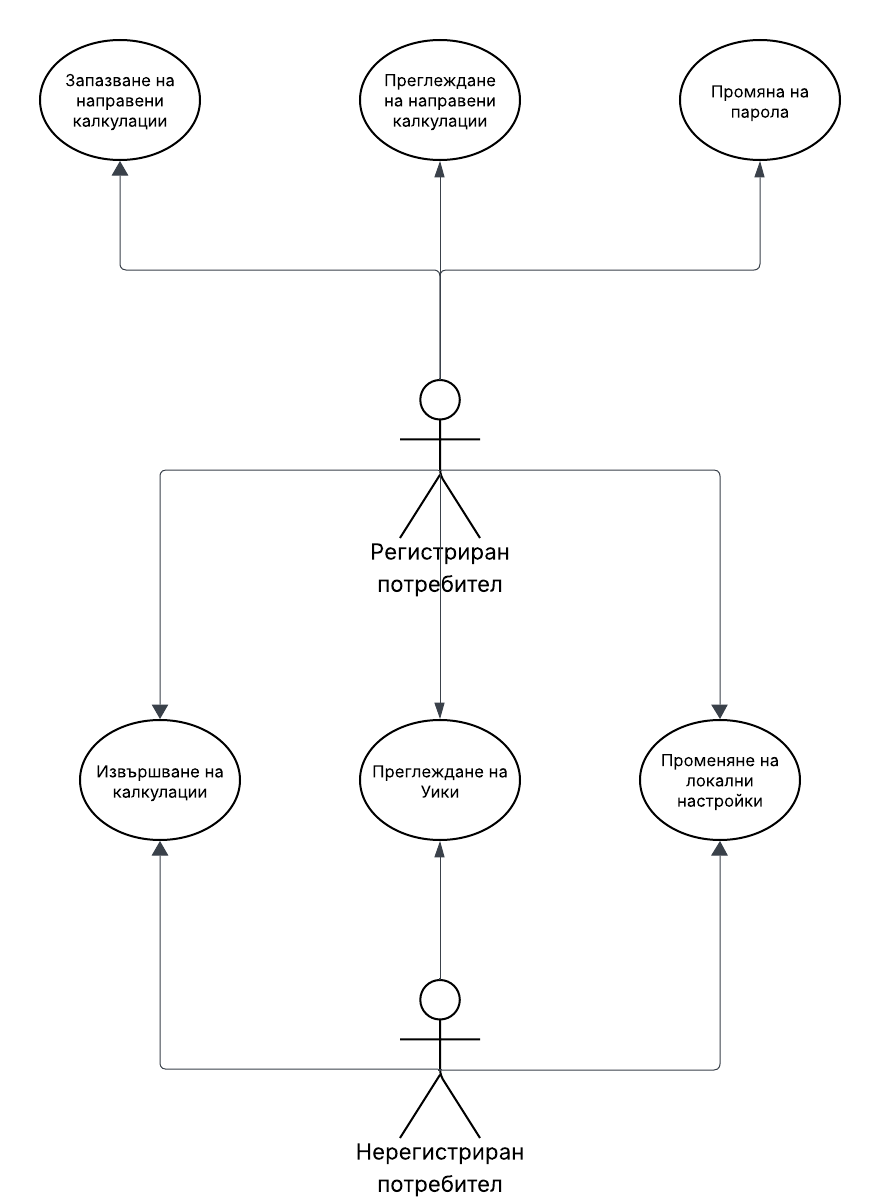
Това е една от най-използваните диаграми, тъй като визуално представя функционалността на системата от гледната точка на потребителя. Главната идея е да се демонстрира взаимодействието на потребителите със системата. Действията се наричат случаи на употреба, а хората или другите взаимодействащи системи се наричат актьори.

Диаграмата на случаите на употреба на разработваното приложение е показана на Фигура 6.

Основните потребители на системата са два на брой:

**Нерегистриран потребител** – потребител, който няма създаден профил в приложението. Има достъп до част от функционалностите на системата, без необходимост от вход. Той може да извършва основни калкулации, както и да разглежда наличната уики информация в системата. Също така има възможност да променя локалните настройки според своите предпочитания.

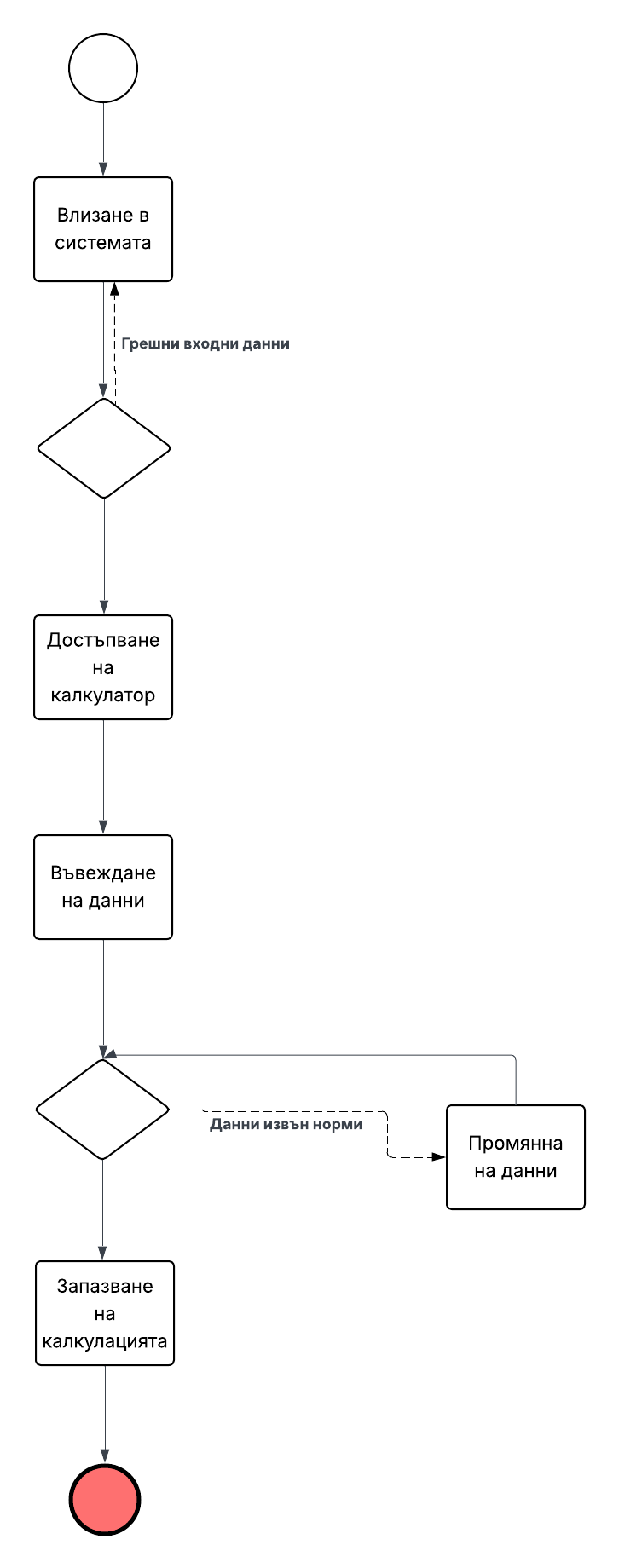
**Регистриран потребител** – вече създал профил и автентикиран в системата. Има разширен достъп до функционалностите, включително всички налични за нерегистриран потребител. Освен това, той може да запазва направените от него калкулации, да ги преглежда по-късно, както и да променя своята парола. Също така има възможност за настройване на локални параметри според личните нужди (език, мерни единици и др.).



Фигура Диаграма на случаите на употреба

### **3.3.2 Диаграми на дейностите – Activity Diagrams**

Това е друг тип UML диаграма, която се използва, за да се опишат работните процеси или дейности използвани в системата. Тя представя визуално последователността на действия и решения взети в даден процес



Фигура Диаграма на дейностите – създаване на нова специалност

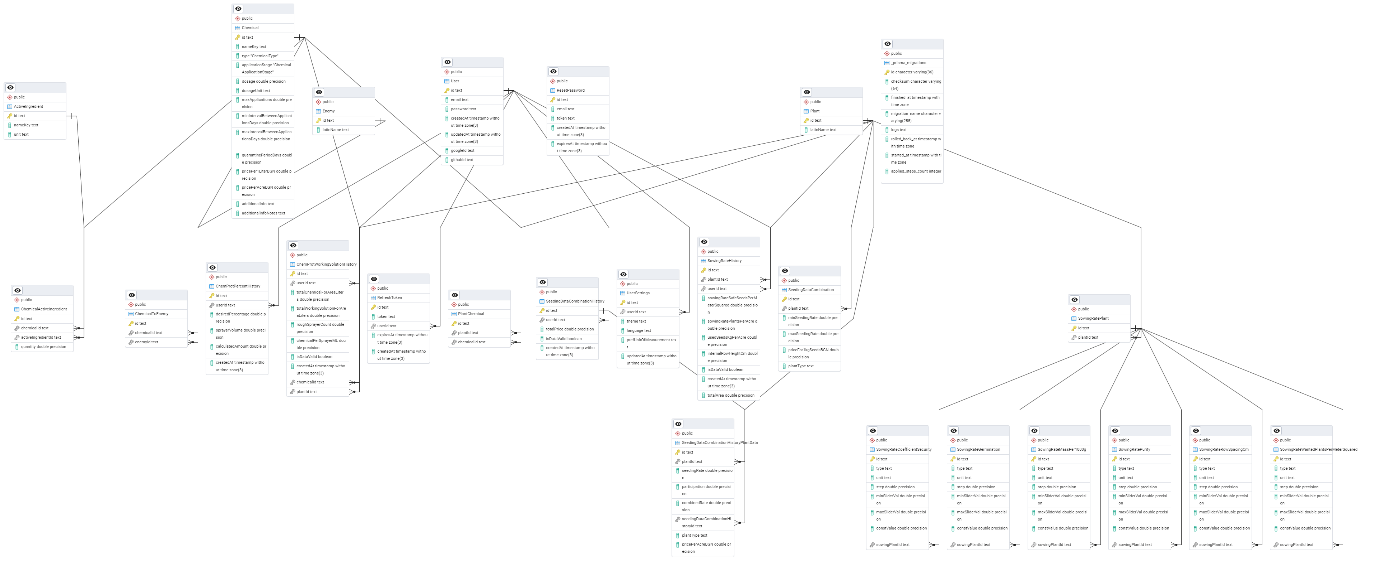
Процесът представен на Фигура 7. се отнася за основната функционалност –извършване на калкулации и тяхното запазване.

# **4. Описание на базата данни**

За реализацията на системата е използвана релационна база от данни. Използвана е СУБД (RDBMS) PostgreSQL. pgAdmin4 е използван като среда за визуализиране на таблиците с от време на време използване и на DBeaver.

## **4.1 Релационен модел на базата данни**

На Фигура 8 е показан релационния модел на проектираната база данни, с таблиците и техните връзки.



Фигура . Релационен модел на базата данни

**4.1.1. Основни таблици**

Базата съдържа няколко вида таблици – за потребителя и автентикация, таблици съдържащи информация за програмата и за записване на потребителско създадена информация.

* Users – Съдържа информация за всеки потребител: имейл, парола (ако не използва OAuth), идентификатори за Google и GitHub, дата на създаване и актуализация. Свързва се с настройки, токени и исторически данни.
* UserSettings – Съдържа настройки за конкретен потребител – предпочитан език, тема и мерна единица. Свързана е едно към едно с таблицата Users.
* RefreshToken – Съдържа уникален токен за обновяване на сесията на потребител, валиден до определена дата.
* ResetPassword – Съдържа токени за заявка за нулиране на парола, валидни за определен период, базирани на имейл адрес.
* Plants – Съдържа информация за всяко растение, включително латинското му име. Свързва се с химикали, данни за сеитба и история на сеитбата.
* SowingRatePlant – Съдържа параметри за изчисление на сеитбена норма за конкретно растение (напр. разстояние между редове, маса на семената и др.).
* SowingRateHistory – Съдържа изчислена история на сеитбени норми за потребители и растения – колко семена на квадратен метър, изразходвани килограми и др.
* SowingRateCoefficientSecurity / Purity / Germination / RowSpacingCm / WantedPlantsPerMeterSquared / MassPer1000g – Подпомагат изчисленията за сеитбената норма, всеки с различна роля. Всяка таблица съдържа информация за типа параметър и свързаните му стойности.
* SeedingDataCombination – Съдържа препоръчителни стойности за минимална и максимална сеитбена норма, цена на семената и вид растение.
* SeedingDataCombinationHistory – История на комбинации от сеитбени данни, избрани от потребителя – включва крайна цена и валидност на въведените стойности.
* SeedingDataCombinationHistoryPlantData – Описва участието на всяко растение в конкретна комбинация – избрана норма, цена и тип растение.
* Chemicals – Съдържа данни за растителни препарати (напр. хербициди, фунгициди), тяхната доза, период на прилагане, карантина и цена.
* ChemicalActiveIngredient – Съдържа съотношението на активните съставки във всеки химикал.
* ActiveIngredient – Списък с всички възможни активни вещества, които могат да присъстват в химикал.
* Enemy – Съдържа вредители (напр. плевели, насекоми), срещу които може да бъде приложен химикал.
* ChemicalToEnemy – Съдържа връзка между конкретен химикал и вредител, срещу който се използва.
* PlantChemical – Свързва конкретни растения с химикали, които могат да бъдат използвани върху тях.
* ChemProtWorkingSolutionHistory – История на изчисления за работен разтвор на химикал спрямо площ, използван обем и дозировка.
* ChemProtPercentHistory – История на изчисления за необходимото количество химикал при зададен процент и обем на пръскачката.

## **4.2. Описание на таблиците и връзките между тях**

## **4.2.1 Таблица Users**

Съдържа информация за регистрираните потребители.

* id – Уникален идентификатор (String)
* email – Имейл на потребителя (String, уникален)
* password – Парола, може да е null (String)
* createdAt – Дата на създаване (DateTime)
* updatedAt – Дата на последна промяна (DateTime)
* googleId / githubId – Идентификатори за OAuth (String, уникални, по избор)
* RefreshToken – Връзка с таблица RefreshToken
* UserSettings – Връзка с таблица UserSettings
* ChemProtPercentHistory / ChemProtWorkingSolutionHistory – История на изчисленията за химическа защита
* SowingRateHistory / SeedingDataCombinationHistory – Исторически сеитбени данни

## **4.2.2 Таблица UserSettings**

Настройки на потребителски профил.

* id – Уникален идентификатор (String)
* userId – Идентификатор на потребителя (String, уникален)
* theme – Тема (light, dark, system)
* language – Език (bg, en)
* prefUnitOfMeasurement – Предпочитани мерни единици (acres, hectares)
* updatedAt – Дата на промяна (DateTime)
* user – Връзка с таблица Users

## **4.2.3 Таблица RefreshToken**

Съдържа токени за обновяване на достъпа.

* id – Уникален идентификатор (String)
* token – Самият токен (String, уникален)
* userId – Идентификатор на потребител (String, уникален)
* expiresAt – Валидност на токена (DateTime)
* createdAt – Дата на създаване (DateTime)
* user – Връзка с таблица Users

## **4.2.4 Таблица ResetPassword**

Таблица за заявка за забравена парола.

* id – Уникален идентификатор (String)
* email – Имейл на потребителя (String)
* token – Уникален токен за смяна на парола (String)
* createdAt / expiresAt – Дати за създаване и изтичане (DateTime)

## **4.2.5 Таблица Plant**

Съдържа информация за растенията.

* id – Уникален идентификатор (String)
* latinName – Латинско наименование (String, уникално)
* plantChemicals – Връзка с PlantChemical
* seedingDataCombination – Препоръчителни сеитбени норми
* SowingRateHistory / SowingRatePlant – История и параметри за сеитба

## **4.2.6 Таблица SowingRatePlant**

Съдържа параметри, нужни за изчисление на сеитбена норма.

* id – Уникален идентификатор (String)
* plantId – Идентификатор на растение (String, уникален)
* rowSpacingCm / germination / purity – Връзки към съответни подтаблици
* plant – Връзка с таблица Plant

## **4.2.7 Таблица SowingRateHistory**

История на изчислени сеитбени норми.

* id – Уникален идентификатор (String)
* userId / plantId – Връзки към User и Plant
* sowingRateSafeSeedsPerMeterSquared / usedSeedsKgPerAcre – Числови стойности
* isDataValid – Валидност на изчислението (Boolean)
* createdAt – Дата (DateTime)
* totalArea – Общата площ (Float)

## **4.2.8 Таблица SeedingDataCombination**

Препоръки за сеитба за дадено растение.

* id / plantId – Уникални идентификатори
* minSeedingRate / maxSeedingRate / priceFor1kgSeedsBGN – Стойности за изчисление
* plantType – Тип на растението (String)
* plant – Връзка с Plant

## **4.2.9 Таблица SeedingDataCombinationHistory**

История на избрани комбинации от сеитбени данни от потребителя.

* id / userId – Уникални идентификатори
* totalPrice / isDataValid – Общи стойности
* createdAt – Дата на създаване
* plants – Връзка с SeedingDataCombinationHistoryPlantData

## **4.2.10 Таблица Chemical**

Съдържа информация за препарати за растителна защита.

* id / nameKey – Уникални идентификатори
* type / applicationStage – Вид и фаза на приложение
* dosage / dosageUnit / pricePer1LiterBGN – Дозировка и цена
* chemicalTargetEnemies / activeIngredients – Връзки към вредители и съставки

## **4.2.11 Таблица ChemicalActiveIngredient**

Съдържа кои активни съставки присъстват в кой препарат.

* id / chemicalId / activeIngredientId – Идентификатори
* quantity – Количество от съставката
* activeIngredient – Връзка с ActiveIngredient
* chemical – Връзка с Chemical

## **4.2.12 Таблица ActiveIngredient**

Списък с всички активни вещества.

* id / nameKey / unit – Основни данни
* chemicals – Връзка с ChemicalActiveIngredient

## **4.2.13 Таблица Enemy**

Съдържа вредителите по растенията.

* id / latinName – Уникален идентификатор и латинско име
* chemicals – Връзка с ChemicalToEnemy

## **4.2.14 Таблица ChemicalToEnemy**

Кои препарати действат срещу кои вредители.

* id / chemicalId / enemyId – Идентификатори
* chemical / enemy – Връзки към Chemical и Enemy

## **4.2.15 Таблица PlantChemical**

Кои препарати са приложими към кои растения.

* id / plantId / chemicalId – Идентификатори
* plant / chemical – Връзки с Plant и Chemical

## **4.2.16 Таблица ChemProtWorkingSolutionHistory**

Изчисления за работен разтвор на химикал.

* id / userId / chemicalId / plantId – Идентификатори
* totalChemicalForAreaLiters / roughSprayerCount – Стойности за изчисление
* isDataValid / createdAt – Флаг за валидност и дата

## **4.2.17 Таблица ChemProtPercentHistory**

Изчисление на обем препарат по процентно съдържание.

* id / userId – Идентификатори
* desiredPercentage / sprayerVolume / calculatedAmount – Стойности
* createdAt – Дата

# **5. Програмна реализация**

Проектът се състой от стандартна структура на NextJS проект с app директория – работа с backend и frontend в един проект с дефиниране на място за извършване чрез декларация във файла. Както по-горе споменато всички request към приложението първо минават през middleware за обработка, главно сигурност и автентикация.

NextJS позволява маршрутизиране на React проект без дефиниция във файл а чрез файловата структура в app директорията. Като се започне от app което отговаря на / всяка папка добавя към маршрута като се определя дали е страница като се използва page.js/.ts и дали е API (Application Programming Interface - Интерфейс за програмиране на приложения) с route.js/.ts. Като js е разширения за файлове от Javascript и .ts за файлове от неговото надграждане Typescript. Това може да бъде лесно видяно на Фигура 9 A-D. Друг аспект на NextJS са динамичните пътища най-лесно видни в app/wiki/sowing/plant/[plantId]. С цел елиминиране на повтарящ се код, лесната работа на програмиста и ясно на пръв поглед разбиране на какво прави се използват динамични пътища. В този случай plantId в скоби което означава че ще бъде заместено от някаква променлива динамично, след това тази променлива може да бъде извикана в кода за обработка с помощта на useParams куката.

|  |  |
| --- | --- |
| Фигура 9 A - Файлова структура | Фигура 9 B - Файлова структура |

|  |  |
| --- | --- |
| Фигура 9 C - Файлова структура | Фигура 9 D - Файлова структура |

Има още много файлове извън тази папка. Както е обяснено по-горе ако файла не отговаря на тези точни правила той не може да бъде достъпен като страница. Това позволява на разпределение на кода в няколко файла и компонента който са независим, позволяващ по-лесно пре използване и тестване. В папката components се намират не зависимите компоненти който биват викана в страници и биват рисувани.

В папката hooks, Фигура х3, се намират куки. В React куките се използват с цел разделяне на логиката от компонента който обработва визуалната част, пре използване на логика и пазене на състояние на отделно място. Добър пример е useSowingRateForm страницата за sowing и нейните компоненти се занимават само и единствено с визуалното обработване на информацията докато логиката като сметки, състояние, и комуникация с API се извършва от куката. Друг добър пример е useTranslate, отново вместо да се взима глобално състояние от redux един компонент не трябва да се занимава с такива неща, стига куката да бъде извикана и се подава реален ключ за превод той ще бъде правилно обработен.

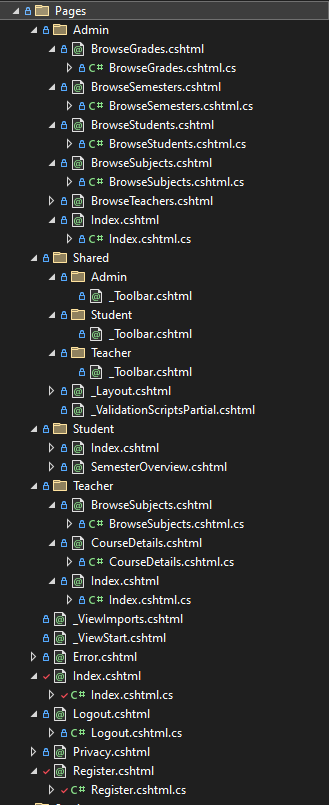
Други важни папки са prisma – съдържа програмно дефинираната структура на базата от данни и начин за връзка с нея. Благодарения на тази дефиниция и Typescript напълно се елиминират грешни SQL команди и грешки при изпълняване на CRUD операции (Create, Read, Update, Delete - Създаване, четене, актуализиране, изтриване). Store – където се съхранява дефиницията на локалните настройки като тема и език и локалното състояние на потребителя. Test-utils – там се съхранява преизползваем код за инструмента за тестване jest, вместо всеки път да се имплементира фалшив redux за тестване кода вече съществува и се вика. Lib – тук се намират различни удобни неща като logger който прави запис на в конзолата със допълнителна информация като точна пътека, колекция от функции като за работа с prisma или за математика.

## **5.1 Layout и Компонентно Дърво**

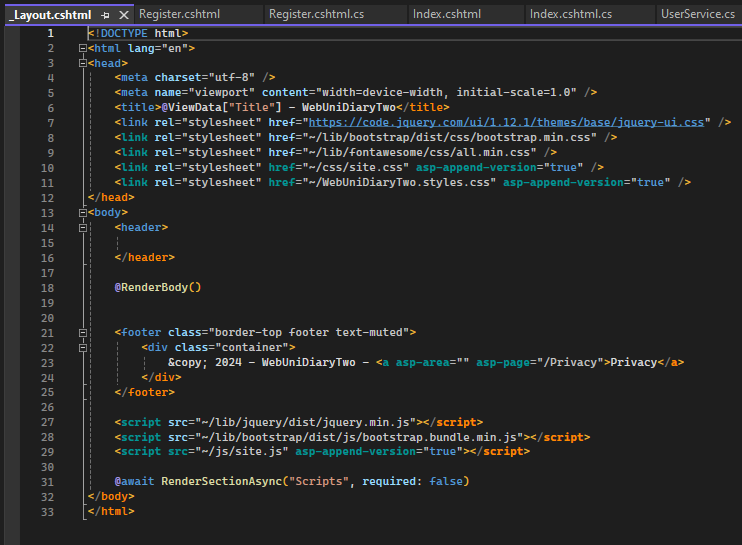
С цел намаляване пренаписване на код и за да има еднакъв приложението има layout.tsx файл който може да се види на Фигура х2. В него може да се видят няколко неща. Код който оправя локалното състояние на автентикация след използване на акаунт от друг източник, създаването на локалното състояние или магазин от react-redux, създаването на механизма който го запазва в браузъра за да не бъде загубено това състояние след презареждане на страница, създаването на системата глобално отговаряща за темата на системата.

В React това се нарича компонентно дърво. Както в HTML и например div всеки компонент се представя под тази форма. Той притежава две главни свойства – props и children. Props са параметри подадени на този елемент, в този пример това ще са initialAuthState на Providers или defaultTheme на ThemeProvider. Те могат да бъдат обработени вътрешно и използвани от компонента, те също биват предадени на всички деца. Children или деца са елементите под определен в дървото, в този пример Header е дете на ThemeProvider и за това получава параметри от него. Плюс на тази архитектура е че стига някъде нагоре в дървото да е добавена информация тя е достъпна, минуса е че обмяната на информация от дете до родител е трудна и за това се използва redux за глобален обект с информация който може да бъде достъпен от всякъде в дървото.

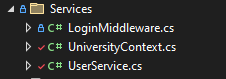
## **5.2 Важни файлове**

Също така там се обработва визията на страницата добавянето на Header + Body + Footer и също така зависимостите на Bootstrap, Ajax, Fontawesome и други. Може да се види на Фигура 5.2.

*Фигура 5.1 – Всички Страници и тяхната йерархия*



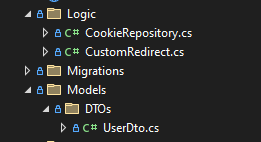
*Фигура 5.2 – Template \_Layout.cshthml*

Освен страниците, приложението притежава и различни услуги. Те се намират в папка Services това са LoginMiddleware, UniversityContext (Db relation) и UserService. Приложени са на Фигура 5.3.

*Фигура 5.3 – Раздел Services*

Препоръчително е всеки вид обработка на информация да бъде изнесен в отделен Service, за да се предотвратят проблеми в приложението и при нужда от бъдещи промени на дадена логика, тя да функционира по един и същ начин навсякъде. Засега само най-базовата логика е изнесена.

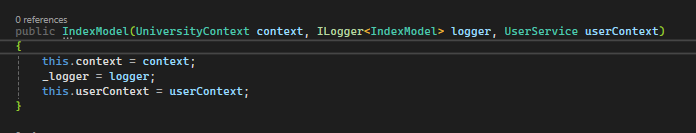
Допълнително, разработката разполага само с един модел DTO (Data Transfer Object), и това е UserDto.cs. Планира се всеки обект да има свой собствен модел за пренос на данни, чрез който да се прилагат проверки и ограничения за това какво може да се извършва с него.

Част от помощната логика на приложението е изнесена в директорията Logic/, където се намират CookieRepository.cs и CustomRedirect.cs. Тези два обекта обработват бисквитките, запазвани в браузъра, и пренасочването на регистрираните потребители. Например, ако вече сме логнати, няма нужда localhost/Register да ни насочва към формата за регистрация.

*Фигура 5.4 – Допълнителната логика и Моделите за DTOs*

## **5.1 Инжектиране на зависимости в Razor Pages (Dependency Injection)**

Инжектирането на зависимости (Dependency Injection, DI) е важен модел за управление на софтуерните зависимости, който улеснява изграждането на гъвкав и проверим код. В контекста на Razor Pages в ASP.NET Core, DI се използва широко за предоставяне на услуги и компоненти на страниците и техните модели (Page Models), без да е необходимо те сами да създават тези зависимости.



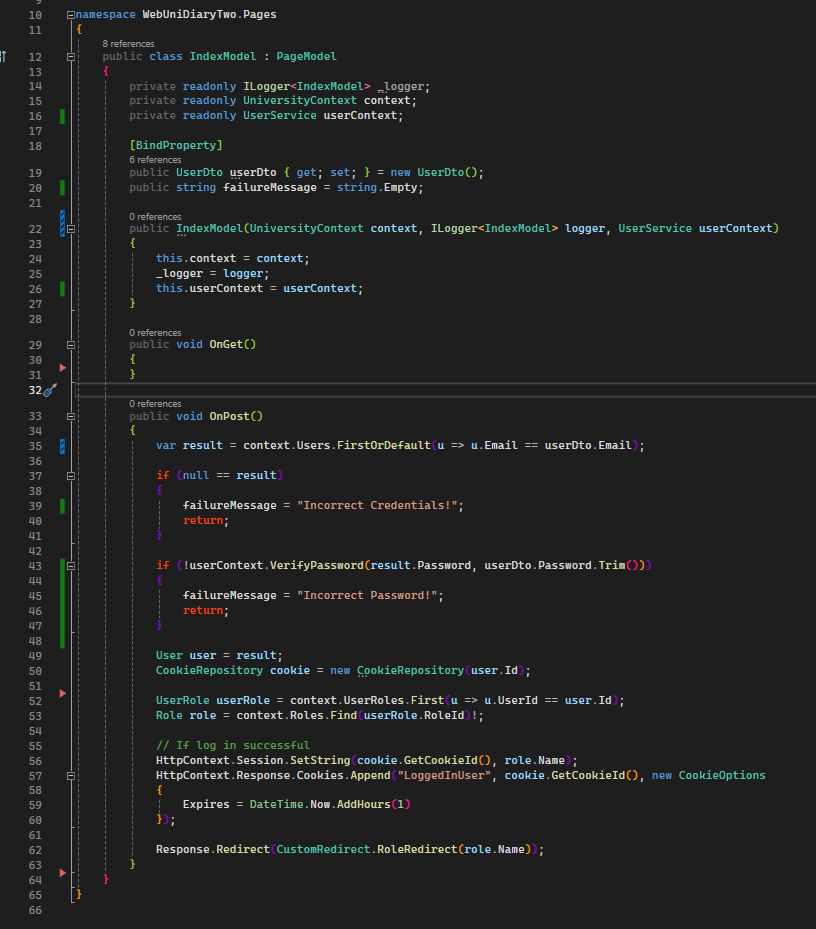
*Фигура 5.5 – Начална на приложението и нейният конструктор.*

На горната фигура е демонстрирано как чрез конструктора се добавя връзката с базата данни UniversityContext, обработката на данни, свързани с потребителя, чрез UserService и интерфейсът ILogger, използван за извеждане на логове в конзолата на сървъра.

## **5.2 Основни страници**

### **5.2.1 Начална страница (Index)**

Използва се за показване на началната страница, където потребителите имат опцията да се впишат в приложението или да се регистрират.



*Фигура 5.6 – Как точно изглежда една .cshtml.cs страница*

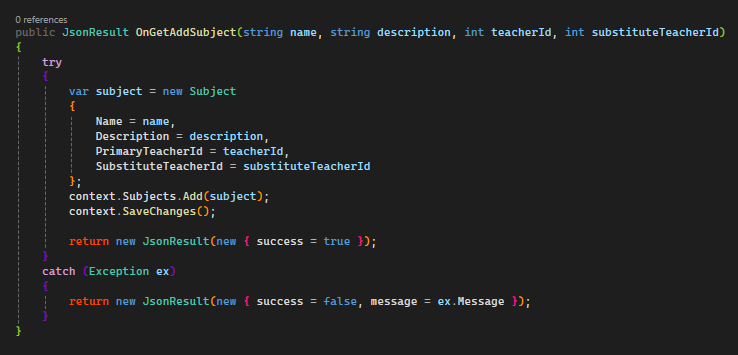
Всяка Razor страница съдържа методи като OnGet() и OnPost(), които са известни като Handler Methods. Те отговарят за това какво се случва, когато страницата се заявява (GET) или когато изпращаме данни към контролера на страницата (POST).

OnGet(): Изпълнява се, когато страницата се зарежда чрез GET заявка. Обикновено тук се инициализират данни, които трябва да бъдат показани на страницата.

OnPost(): Изпълнява се, когато потребителят изпрати данни чрез формуляр на страницата (POST заявка). Обикновено тук се обработват и валидират изпратените данни.

Освен стандартните OnGet() и OnPost() методи, всяка Razor Page може да поддържа и именовани Named Handlers, като например OnGetAddSubject(). Това са разширени методи, които се използват за обработка на специфични действия или събития, често активирани чрез AJAX заявки или допълнителни действия на потребителя.

Пример за разширен Handler е показан във Фигура 5.7. Той е OnGet() хандлър със специфично име AddSubject.



*Фигура 5.7 – Разширен Handler – за допълнителна логика.*

Как се вика този хандлър -> url: '@Url.Page("/Admin/BrowseSubjects", new { handler = "AddSubject" })'

Както се забелязва, използван е OnGet(), за да се изпратят данни, но реално трябва да е OnPost(). Поради неясна грешка, Ajax заявката не се обработваше от приложението, което наложи употребата на OnGet() handler. В бъдещи версии това ще бъде коригирано.

Относно Index() страницата: Тя служи като главна страница и помага на потребителите да се впишат в системата. Използва DTO (data transfer object) - userDto, за да валидира данните, и след успешна валидация потребителят се запазва в сесията и му се създава бисквитка, за да остане вписан. Още една област за подобрение е, че CookieRepository няма Factory обект, който да генерира бисквитката. В момента това действие е директно записано в самото репозитори.

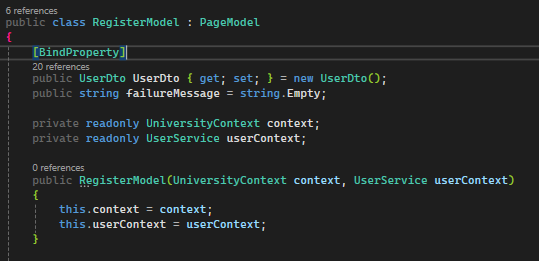
### **5.2.2 Страница Register**

На тази страница се обработват данни, свързани с регистрацията на нов потребител, и се проверява дали потребителската роля съществува в базата данни. Ако не съществува, тя се добавя. Хеширането на пароли се извършва чрез PasswordHasher, наличен в библиотеката Microsoft.AspNetCore.Identity. Накратко, използва се хеширащият алгоритъм PBKDF2, който прави разбиването на хеша много трудно, като хеширането е 256-битово.

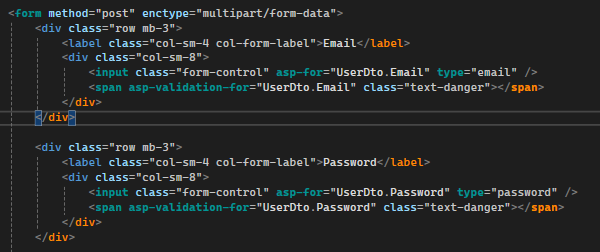
След успешната регистрация на нов потребител, контролерът генерира нова бисквитка за този потребител и я запазва в сесията на сървъра. По този начин потребителят автоматично се вписва в системата и се пренасочва към началната страница според своята роля.

Използва се само методът public void OnPost().

По-рано в началната страница беше споменато, че приложението използва UserDto. Тук ще разгледаме как чрез [BindProperty] регистрационната страница валидира потребителската заявка.



*Фигура 5.8 – Страница Register контролер*

**

*Фигура 5.9 – Страница Register html*

Фигури 5.8 и 5.9 демонстрират как в C# се обявява, че даден обект ще валидира своите променливи и при проблем <span asp-validation-for> ще съдържа съответната стойност. Това се случва след изпращането на заявката, и ако възникне проблем, потребителят ще бъде пренасочен обратно към същата страница.

### **5.2.3 Страница Logout**

Тази страница се използва единствено за отписването на вписан потребител от сесията му. Бисквитката с неговия уникален ключ за достъп се изтрива от браузъра, след което той се премахва от сесията на сървъра.

Страницата няма визуална част, а потребителят винаги се пренасочва към началната страница чрез Response.Redirect("/"). В бъдеще е планирано да се запазва входният достъп на потребителя в базата данни, така че дори при рестартиране на сървъра, ако браузърът съдържа бисквитка, потребителят да бъде вписан автоматично.

### **5.2.4 Страница Admin – BrowseGrades**

Това е страницата, на която се показват всички оценки на всички студенти за определен предмет и специалност. За да се реализира това, както е показано на Фигура 5.10, трябва да се извлекат данните за конкретната дисциплина на дадена специалност, както и да се намерят оценките за тази дисциплина. Използва се eager loading, за да може с една заявка да се съберат всички нужни данни:

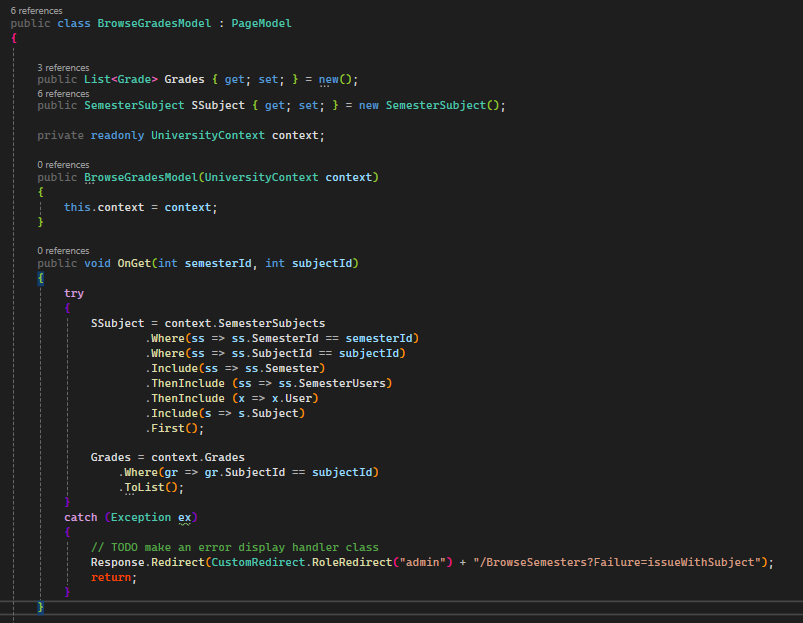
* Данни за специалността (Semester)
* Данни за дисциплината (Subject)
* Данните за студентите (User)

По този начин, чрез вече дефинираните връзки в ERM, C# обектите могат логически да се свържат и да се заредят.

При проблем или липса на данни, администраторът ще бъде върнат на предишната страница.

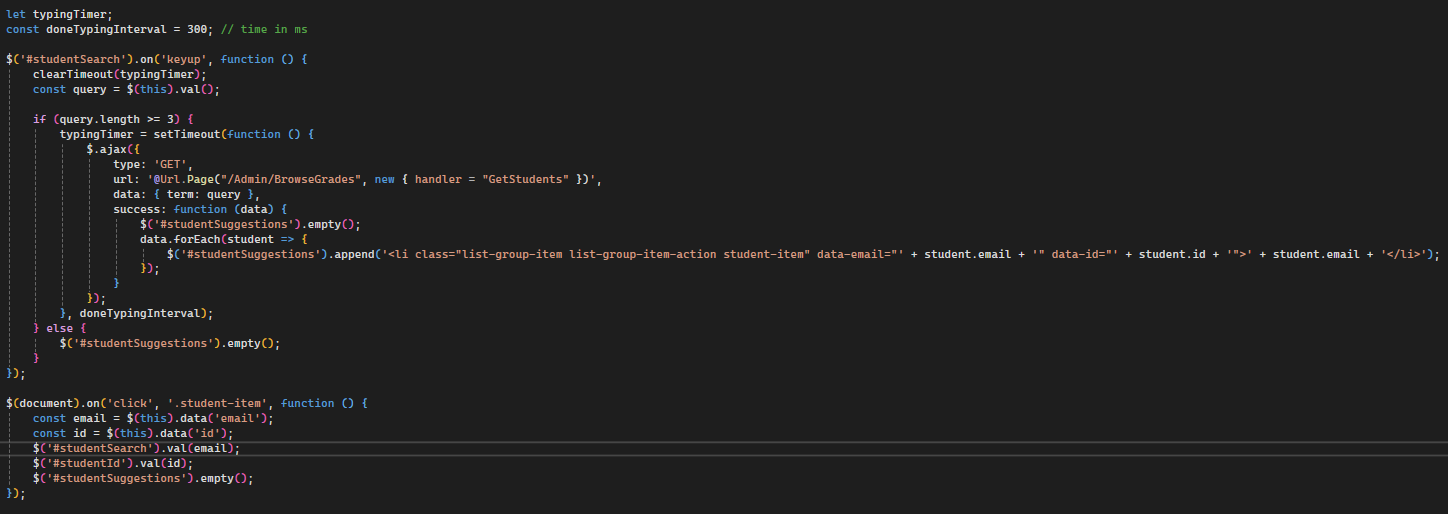
На тази страница се поддържа серия от допълнителни AJAX handlers.

1. AddGrades – IActionResult метод, който при подаване на семестъра, дисциплина, оценката и типа на оценката, ще направи опит да я запази в базата данни. При успех ще презаредим страницата за да се визуализира новата оценка.

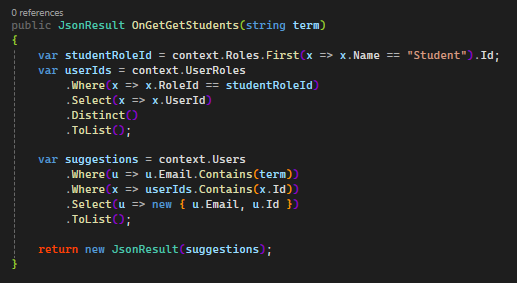


*Фигура 5.10 – Страница BrowseGrades използваща eager loading*

1. AddStudent – Метод, с който лесно можем да добавим нов студент към специалността. Отново е реализиран чрез IActionResult и изисква подаването на студентския номер и семестъра на студента. Извършват се серии от проверки, преди студентът да бъде добавен към специалността. Първо, потребителят трябва да е с роля "студент", както и студентът да не присъства вече в тази специалност.
2. GetStudent – Този метод предоставя списък с имейл адреси на студенти в реално време, докато се попълва формата за добавяне на нов студент, което улеснява администраторът при избор на правилния потребител. Методът връща JsonResult. На Фигури 5.11.1 и 5.11.2 е показано как работи тази функционалност. Чрез HTML append старите данни се изтриват и се заменят с нови имейл адреси. В момента заявки се изпращат само ако потребителят е написал повече от 3 букви, а интервалът между заявките е над 1/3 от секундата. След това потребителят може да избере правилния студент чрез кликване с мишката.



*Фигура 5.11.1 – JS ajax метод за попълване на потребители, когато се търси чрез имейл адрес*

**

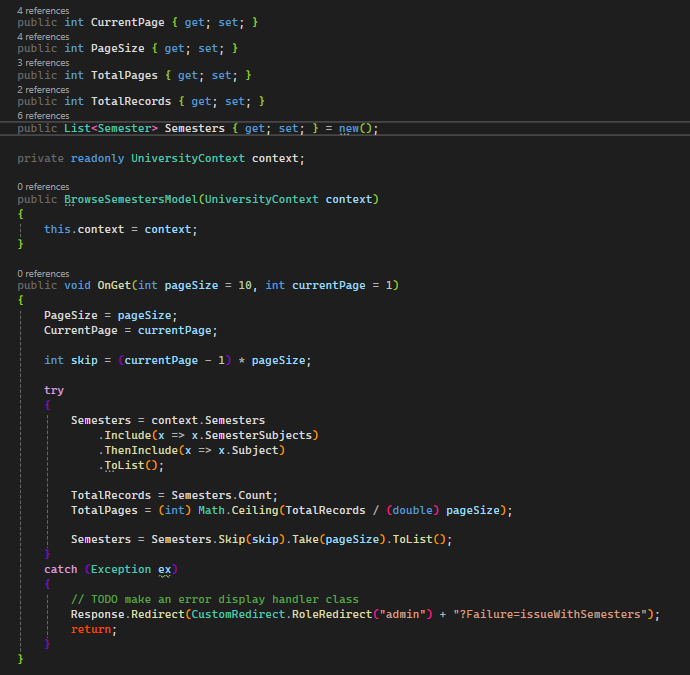
*Фигура 5.11.2 – C# частта която отговаря за обработването на ajax request*

### **5.2.5 Страница Admin - BrowseSemesters**

На тази страница администраторите на приложението могат да видят всяка добавена специалност. Тук те могат да добавят нови специалности, да добавят предмети към тях или да променят данни по вече съществуващи специалности.

Контролерът на страницата притежава един OnGet() метод, който филтрира и организира страниците на списъка със специалности. Използването на тази техника позволява разделянето на голям обем данни на по-малки части, разпределени на множество страници. Потребителят има възможността да определи броя на показваните данни на екрана. Това се реализира чрез следене на текущата страница, общия брой данни, възможни за показване на екрана, броя на необходимите страници за показване на всички данни, както и общия брой записи.

След като разполагаме с тази информация, можем да филтрираме заявката към базата данни, за да получим само нужните данни. На Фигура 5.12 е демонстриран OnGet() методът на този контролер. Следващите страници, които използват странициране, следват същата логика.



*Фигура 5.12 – Странициране на BrowseSemesters*

Освен това, чрез AJAX заявки е имплементирана възможността за добавяне на нова специалност. Методът AddSemester връща IActionResult и очаква име на специалността и продължителност, подадена като брой на семестрите. Не е възможно добавянето на нечетен брой семестри. При успех страницата се презарежда.

Други два AJAX метода са тези за редактиране на специалност и вземане на данни за специалност, които работят заедно. GetSemester и EditSemester връщат IActionResult. При промяна на броя на семестрите се обновява и крайната дата на специалността.

Последните два AJAX метода на тази страница са свързани с добавянето на нова дисциплина към даден семестър на специалност. AddSubject служи за добавянето на новата дисциплина и връща IActionResult, докато методът GetSubject подпомага autocomplete функцията на формата. Когато потребителят въведе 3 букви от името на дисциплината, се прави заявка към базата данни и се връщат всички дисциплини, съдържащи тази комбинация.

### **5.2.6 Страница Admin – BrowseStudents**

Използва се, за да се покажат всички студенти, регистрирани в приложението. Контролерът има OnGet() метод и няколко AJAX метода.

Както при предишните страници, в OnGet() се реализира логиката, свързана със страницирането на страниците. Първо се събират всички потребители със студентска роля, след което те се разделят според лимита на записи, които могат да се видят на една страница.

От тази страница, освен разглеждането на всички студенти, може да се добавя нов студент чрез AJAX метода AddStudent, който е от тип IActionResult. Важно е, че имейлът винаги трябва да бъде уникален, за да се добави нов запис.

Друга функционалност е промяната на статуса на студент – дали е активен или не. Това се извършва чрез AJAX метода ToggleStatus, като се изпращат потребителският номер и флагът дали новият статус е true или false. Методът връща IActionResult.

Последната основна функционалност е промяната на данни на студент. Първо, данните на даден потребител се попълват във формата чрез AJAX заявка GetStudent, а след това, при изпращане на промените, те се обработват от EditStudent. И двата метода са от тип IActionResult.

### **5.2.7 Страница Admin - BrowseSubjects**

Тази страница се използва, за да се покажат всички налични предмети, добавени в приложението. Контролерът на страницата предлага OnGet() метод, който подготвя данните за страницираенто, както при предишните страници. Освен заявката за предметите, тук се прави и такава за подготовка на данните за преподавателите, свързани с тези предмети.

Контролерът също така предоставя възможност чрез AJAX за добавяне на нови предмети, редактиране на съществуващи предмети, автоматично предсказване на имейл адресите на преподавателите при добавяне на нов предмет или редактиране на съществуващ, както и AJAX заявка за логическа деактивация на дисциплините.

* AddSubject – връща JsonResult – обработва добавянето на нова дисциплина.
* EditSubject – връща IActionResult – отговаря за редактирането на вече съществуваща дисциплина.
* GetSubject – връща IActionResult – връща търсена дисциплина, като се подаде Id.
* ToggleStatus – връща IActionResult – обновява статуса на дисциплината чрез Boolean флаг.
* GetEmail – връща JsonResult – връща потребители отговарящи на даден имейл адрес. Като тези потребители могат да се единсвено преподаватели.

### **5.7.8 Страница Admin - BrowseTeachers**

Страницата се използва, за да се покажат всички преподаватели в приложението. Контролерът отново предлага OnGet() метод, който отговаря за подготвянето на данните на страницата. Също така, чрез AJAX заявки се обработва логиката по добавянето на нов преподавател, редактирането на данни на вече съществуващи преподаватели, както и логическото деактивиране и активиране на акаунтите.

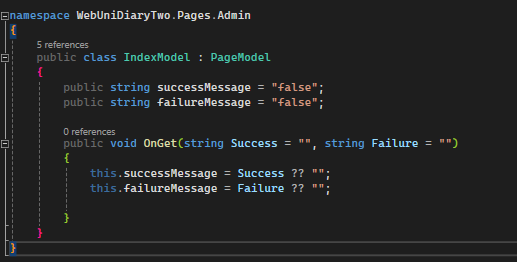
Всички ajax заявки:

* AddTeacher – връща IActionResult - Служи за добавяне на нов преподавател. Имейл адресът трябва да е уникален. При успешно добавяне страницата се презарежда.
* EditTeacher – връща IActionResult – Изпълнява промятната на данни свързани с вече съществуващ преподавател.
* GetTeacher – връща IActionResult – Спомага попълването на данните на формата за редактиране на преподавател
* ToggleStatus – връща IActionResult – обновява логическия статус на преподавателя

### **5.2.9 Страница Admin - Index**

Това е главната страница на админският панел. В нея има линкове по подразбиране за бърз достъп към дадена страница. Тези линкове могат да се конфигурират.

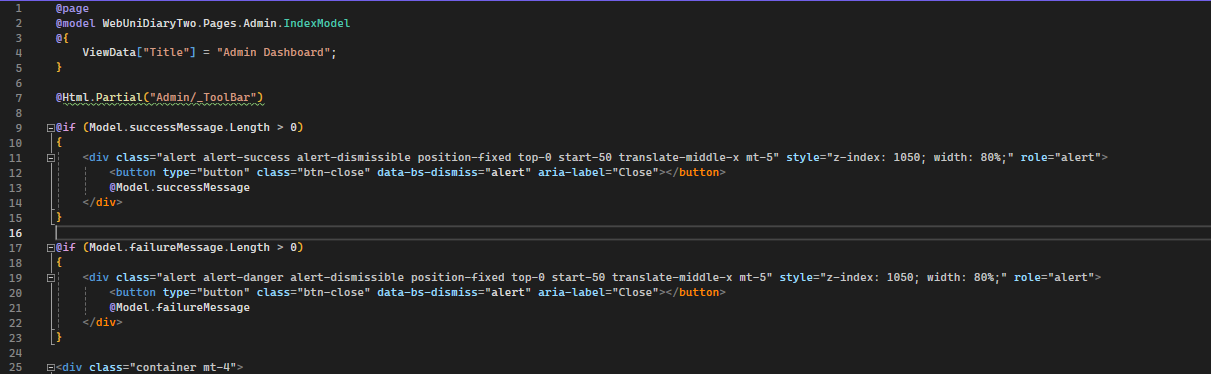
Допълнително главната страница се използва за редирект, при случай че има проблем с приложението. Или е извършено някакво действие успешно, което няма страница за пренасочване.



*Фигура 5.13 – Как изглежда контролера на Admin/Index страницата*

На Фигура 5.13 и 5.14 може да се види как изглежда index страницата. По този начин се извеждат всякакви информационни съобщения в приложението.

Допълнително чрез @Html.Partial() се вмъкват помощните изгледи. В случая на Фигура 5.14 се вижда как се добавя администраторския панел с линковете, в горната част на страницата.



*Фигура 5.14 – Част от Admin/Index страницата*

### **5.2.10 Страница Student - SemesterOverview**

Тази страница се използва от студентите, като тук те могат да видят своите оценки по предметите, които изучават, в зависимост от избраната специалност.

Контролерът има само един OnGet() метод, който приема номера на специалността. След това се извличат данни единствено за съответния потребител и тази специалност. Ако студентът не изучава тази специалност, той ще бъде пренасочен към главната страница.

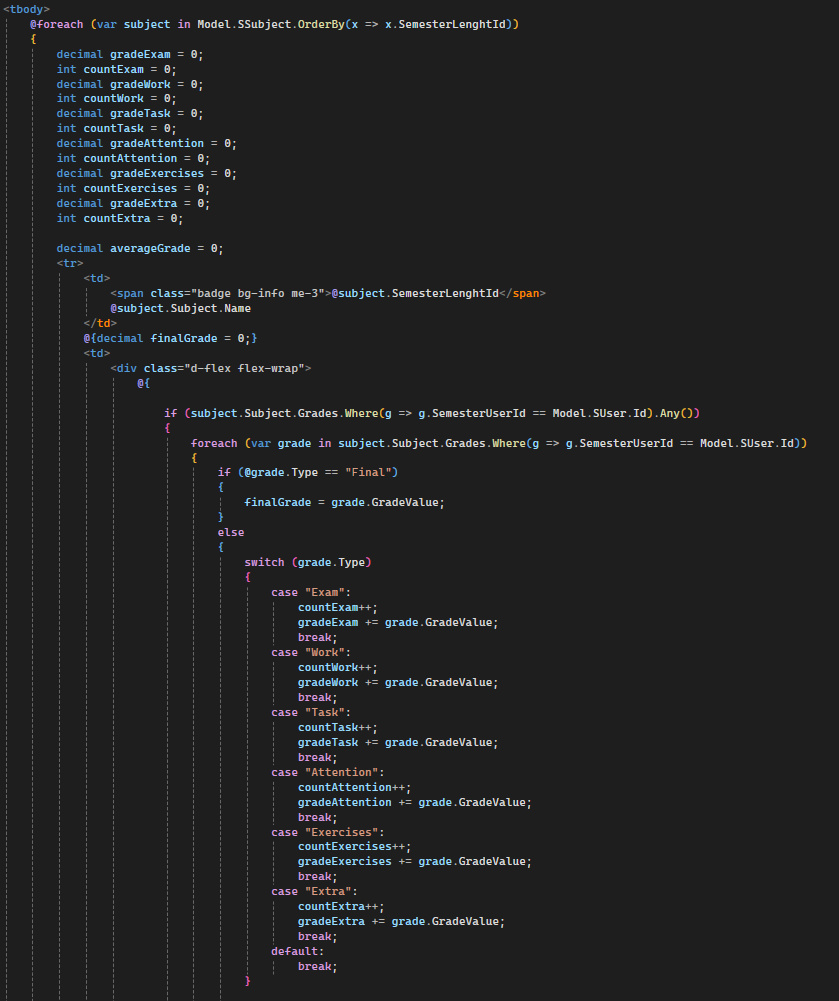
Първо се извличат всички оценки и се инициализират променливите gradeExam, gradeWork, finalGrade и други. При всяко добавяне се брои броят на въведените оценки с помощта на други променливи като countExam, countWork и т.н. След това за всеки тип оценка резултатът се дели на броя оценки, умножава се по тежестта на формулата и се дели на 100.

Как точно работи логиката по изчисляването на препоръчителната крайна оценка по дадена дисциплина. Първо се намират всички оценки за дадена дисциплина за даден студент. След това се филтрират по категории (Примерно оценки от изпит, оценки по задачи и други). Пресмята се индивидуално броят на оценките по групи, после на всяка група се сумира средната оценка, тя се умножава по тежеста на формулата свързана с тази група и накрая се дели на 100. По този начин получаваме успехът за дадена група, който използваме за да получим крайната оценка.

Пример: Има 3 оценки от тип Exam (5, 6, 6). Събираме ги и получаваме 17, след което делим 17 на 3 и получаваме 5,666. Накрая умножаваме по тежестта (примерно 80% от оценката) – 5,666 \* 80 = 453,333, и разделяме на 100. Крайният резултат е 4,53 (след закръгляне до втория знак). По този начин, след като студентът получи и другия тип оценка, оставащите 20% също могат да се добавят и да формират крайния успех по предмета.

На Фигура 5.15 и Фигура 5.16, са показани съответно страниците, на които е приложена тази логика.

В бъдеще е препоръчително логиката за изчисляване да бъде изнесена в Repository, което значително ще улесни повторното използване на тази логика на различни страници.



*Фигура 5.15 – Инициализирането на променливите свързани с успеха*

**

*Фигура 5.16 – Пресмятането на различните оценки и средният успех*

### **5.2.11 Страница Student - Index**

Страницата Student/Index е началната страница за всеки вписан студент. Тя предоставя достъп до преглед на всички специалности, които студентът изучава, както и информация за продължителността на всяка специалност.

Контролерът на страницата съдържа OnGet() метод, който проверява дали логнатият студент има записани специалности, и ако има, показва какво изучава.

### **5.2.12 Страница Teacher – BrowseSubjects**

На тази страница преподавателите могат да видят всички предмети, на които преподават. Контролерът отговаря за OnGet() метод и няколко AJAX заявки.

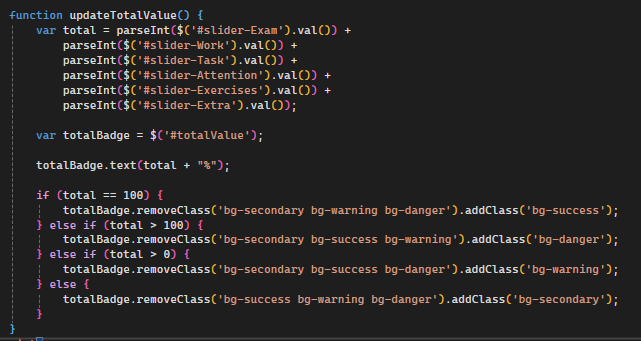
Първо, предметите се сортират по това дали преподавателят е главен или помощен преподавател.

Първата AJAX функция, GetFormula, отговаря за връщането на текущата формула за дадена дисциплина, ако изобщо има такава. Резултатът е IActionResult.

Другият AJAX метод, SaveFormula, служи за запазване на правилно попълнена формула за дадена дисциплина. Важно е, че общият процент на всички полета във формулата трябва да бъде точно 100%. Само тогава резултатът се запазва.

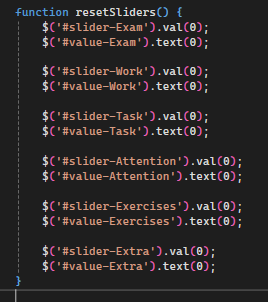
Освен AJAX заявки, на страницата има и няколко помощни JS метода.

Първият метод е за визуално известяване при промяна на стойностите във формата, показващ дали текущата настройка е валидна. Това е илюстрирано на Фигура 5.17.



*Фигура 5.17 – Js метод за визуално известяване на потребителя дали текущата настрайка на формулата е валидна*

Друг помощен метод е resetSliders(), който служи за връщане на стойностите обратно на 0%. Това е необходимо в случаите, когато избраният предмет няма предварително зададени стойности, и формата трябва да се върне към настройките по подразбиране. Методът също позволява на потребителя да нулира данните, ако не е доволен от крайния резултат. Действието на този метод е показано на Фигура 5.18.

*Фигура 5.18 – Js метод, служещ за възстановяване на стойностите на формата*

### **5.2.13 Страница Teacher – CourseDetails**

Това е страницата, където преподавателите могат да нанасят оценки, да проверяват текущия успех на студентите и да въвеждат крайна оценка. След като крайната оценка вече е въведена, не могат да се нанасят нови оценки по тази дисциплина за съответния студент.

Контролерът на страницата предоставя OnGet() метод и два AJAX метода.

За да се визуализира правилно страницата, в OnGet() метода трябва да се подадат правилният номер на семестъра и номерът на дисциплината. След това от базата данни се зареждат информацията за дисциплината, оценките и наличната формула за изчисляване на средния успех по дисциплината. Допълнително, оценките на тази страница се групират по тип и по успех, за улеснение на преподавателя.

Контролерът също така отговаря и за 2 ajax метода:

* SubmitGrade – Тук се обработва запазването на нова оценка, важно е тя да е реално положително число. По-голямо или равно на 2 и по-малко или равно на 6. При всеки друг случай оценката е невалидва.
* SubmitFinalGrade – Тук се запазва финалната оценка по предмета. Отново трябва да отговаря на условията за валидна оценка и чак тогава се запазва в базата данни. Важен момент е, че крайната оценка се въвежда по преценка на преподавателя. Дори средният успех да е 4,5 преподавателя има свободата да нанесе друга оценка различна от 5.

Страницата предлага валидация дали има подготвена формула за дисциплината. И показва визуално ако липсва такава, за да може преподавателят да въведе нова при този случай.

### **5.2.14 Страница Teacher – Index**

Това е началната страница за всички преподаватели. На нея могат да се видят линкове за бърз достъп до други страници, които могат да бъдат конфигурирани от настройките на акаунта.

Контролерът на страницата съдържа OnGet() метод, който подготвя съобщенията по страницата. Страницата е подобна на тази на администраторите.

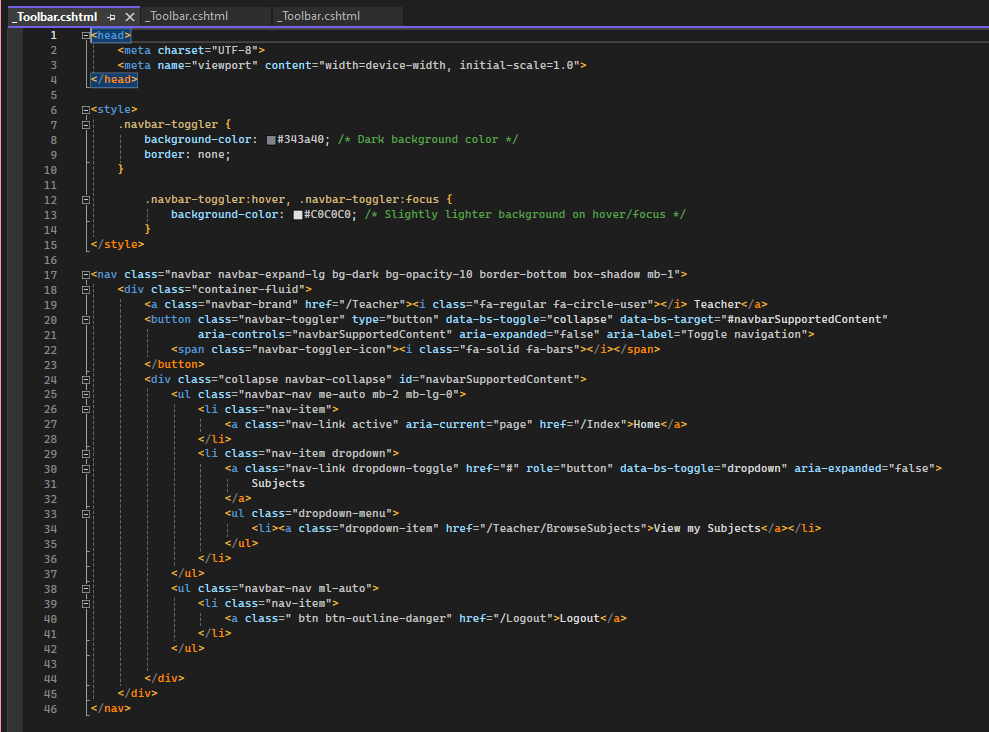
### **5.2.15 Страница Shared**

Тук се съдържат всички споделени HTML темплейти, използвани в другите страници.

На Фигура 5.19 е даден пример за това как изглежда една споделена страница. Вграденият CSS се използва за падащите менюта при избора на линк. Останалата част от HTML е свързана с навигационния панел. По този начин всеки от контролерите получава линкове, които са свързани с неговата функционалност.

Пример за това как се извиква shared view -> @Html.Partial("Teacher/\_ToolBar")

Общо споделените изгледи са 4: Admin/\_Toolbar.cshtml, Teacher/\_Toolbar.cshtml, Student/\_Toolbar.cshtml, \_Layout.cshtml – отговарящ за изгледа на всяка една страница, както бе споменато по-рано.



*Фигура 5.19 – Shared View / Teacher / \_Toolbar.cshtml*

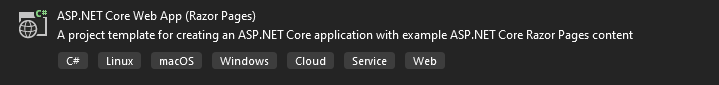
# **6. Ръководство за работа със системата**

## **6.1 Инсталиране на системата.**

### **6.1.1 Нужни ресурси за инсталирането на системата**

За успешна инсталация са нужни следните софтуерни изисквания:

* Visual Studio 2022.
* Добавка ASP.NET Core Web App (Razor Pages) (Фигура 6.1).
* .NET 8.0 – Configured for HTTPS, Do not use top-level statements.
* Microsoft SQL Server.



*Фигура 6.1 – ASP.NET*

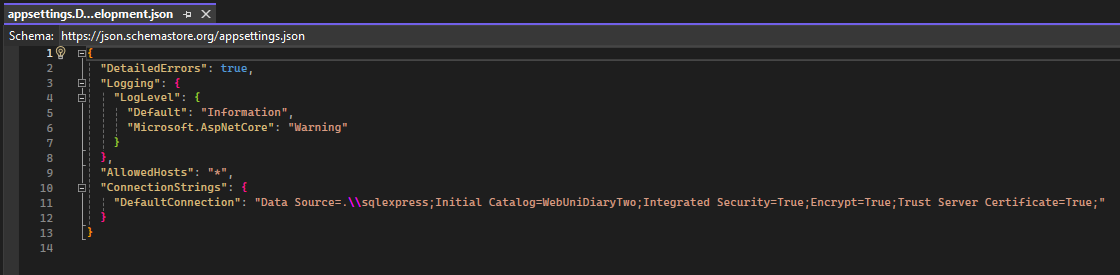
### **6.1.2 Стартиране на проекта**

Първо трябва да се потвърди, че connection стрингът е настроен правилно. Фигура 6.2 показва следния пример:

"DefaultConnection":"Data Source=.\\sqlexpress;Initial Catalog=WebUniDiaryTwo; Integrated Security=True;Encrypt=True;Trust Server Certificate=True;"

След това, при нужда, трябва да се конфигурира SQL Server, като задължително версията му трябва да бъде над 2014, тъй като eager loading е достъпен от тази версия нагоре.

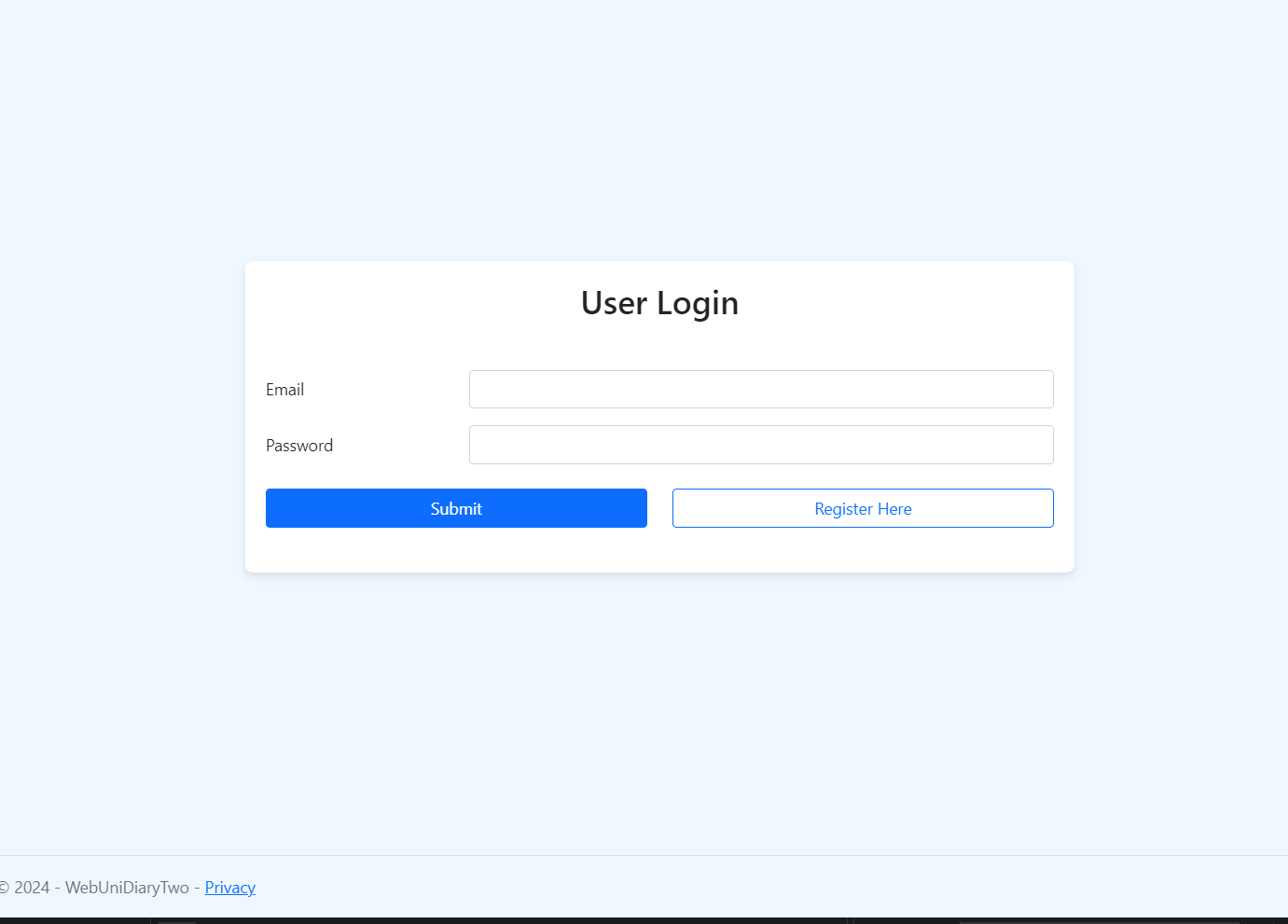
При първото стартиране на SQL Server, трябва да се добави потребител от всеки тип (администратор, преподавател и студент), за да могат потребителските групи да бъдат създадени.



*Фигура 6.2 – Конфигурационни данни*

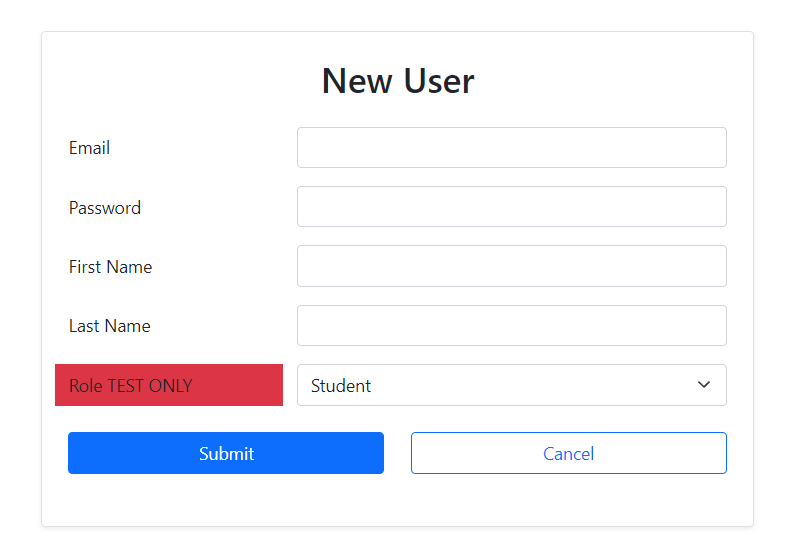
След това може да се стартира поректа. При успех ще се покаже Log In страницата след като се стартира преложението. Фигура 6.3 показва точно това.

Тук потребителите имат опцията да влязат в собственият си акаунт или да се регистрират.



*Фигура 6.3 – Начална страница на системата*

## **6.2 Регистрация на нови потребители**

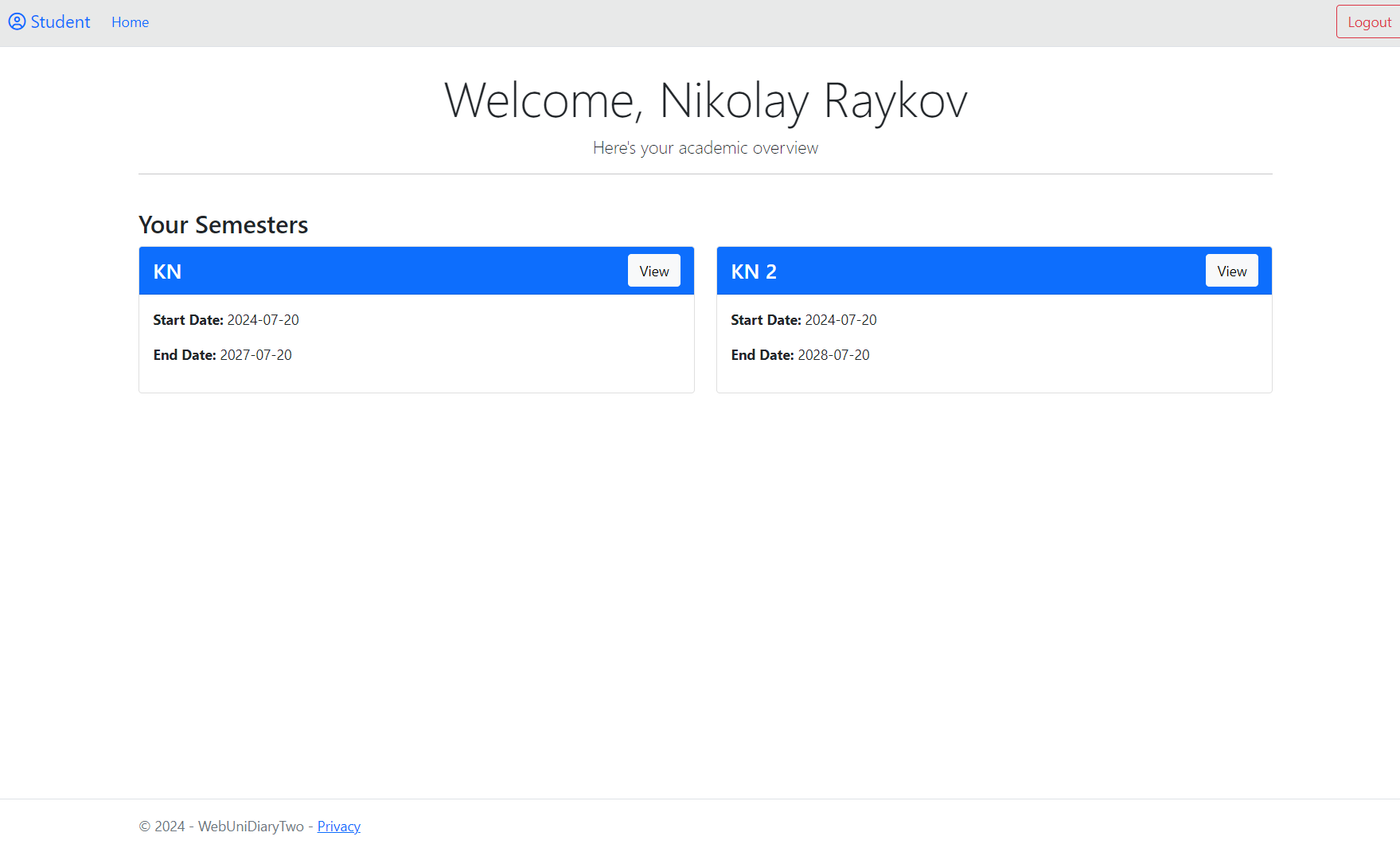


*Фигура 6.4 – Регистрационната форма*

Регистрирането на нови потребители се извършва след избиране на опцията Register Here, както е показано на Фигура 6.3. След това ще се зареди страницата, показана на Фигура 6.4, където могат да се попълнят данните и да се създаде нов студент. Допълнително, за целите на демонстрацията, от тук могат да се добавят също преподаватели и администратори.

### **6.3 Употреба на приложението като студент**

След успешно вписване в платофомата (Фигура 6.3) всеки студент ще бъде приветсван от главната страница на контролера.

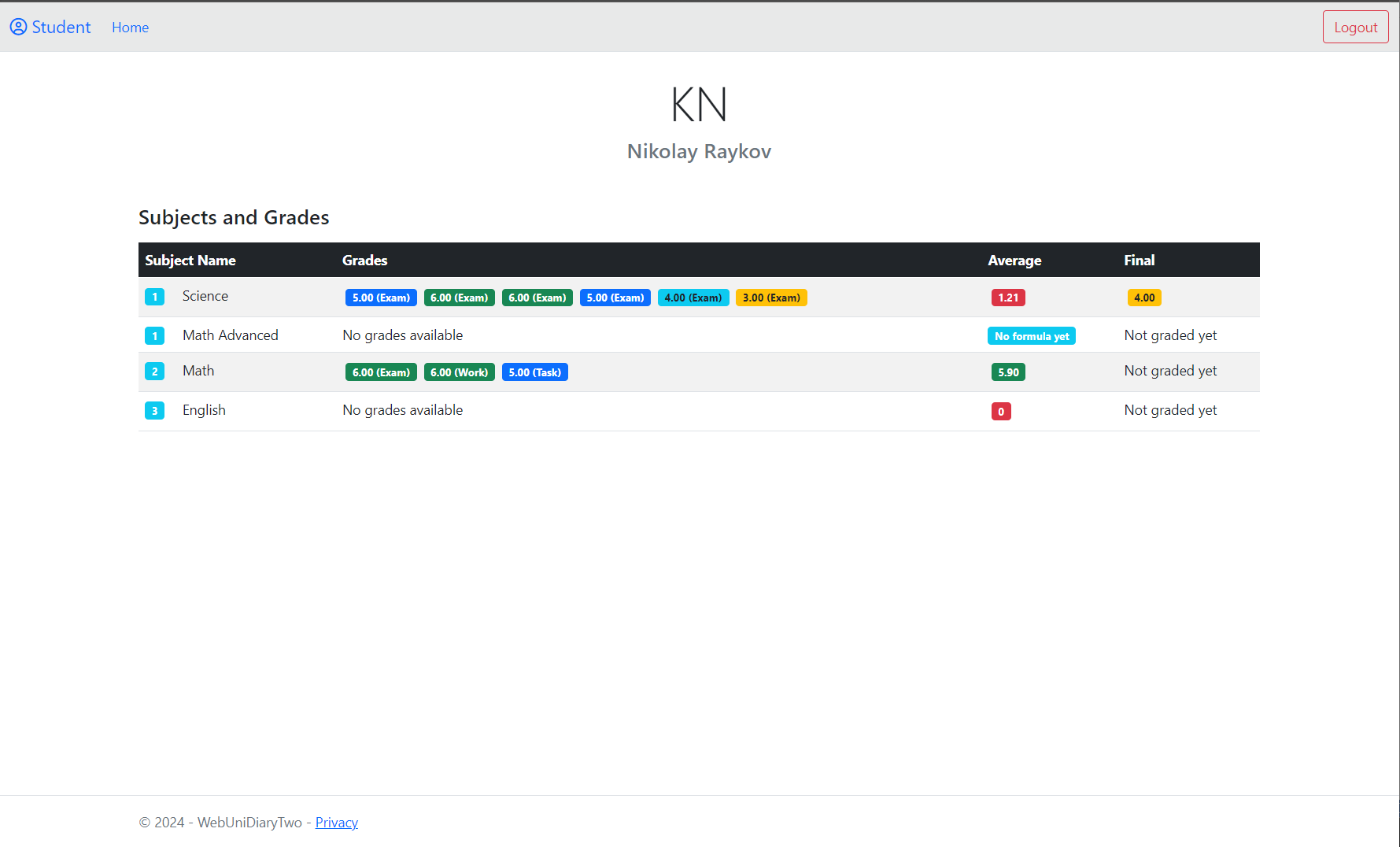


*Фигура 6.5 – Успешно вписан студент, страница Student/Index*

Тук всеки студент може да избере и разгледа своя учебен план или да прегледа профила си в приложението.

Също така приеложението поддържа опцията за изучаване на множество специалности от един студент. При избор на специалност се показват всички дисциплини, текущият успех, средната пресметната оценка по дисциплината и, ако преподавателят е оценил студента, крайната оценка.

Както е показано на Фигура 6.6, може да се види страницата и различни примерни ситуации: как изглежда дисциплина с крайна оценка, пресмятането на средната оценка по формула, както и случаите, в които студентът няма нанесени оценки или дисциплината няма зададена формула за пресмятане на крайната оценка.

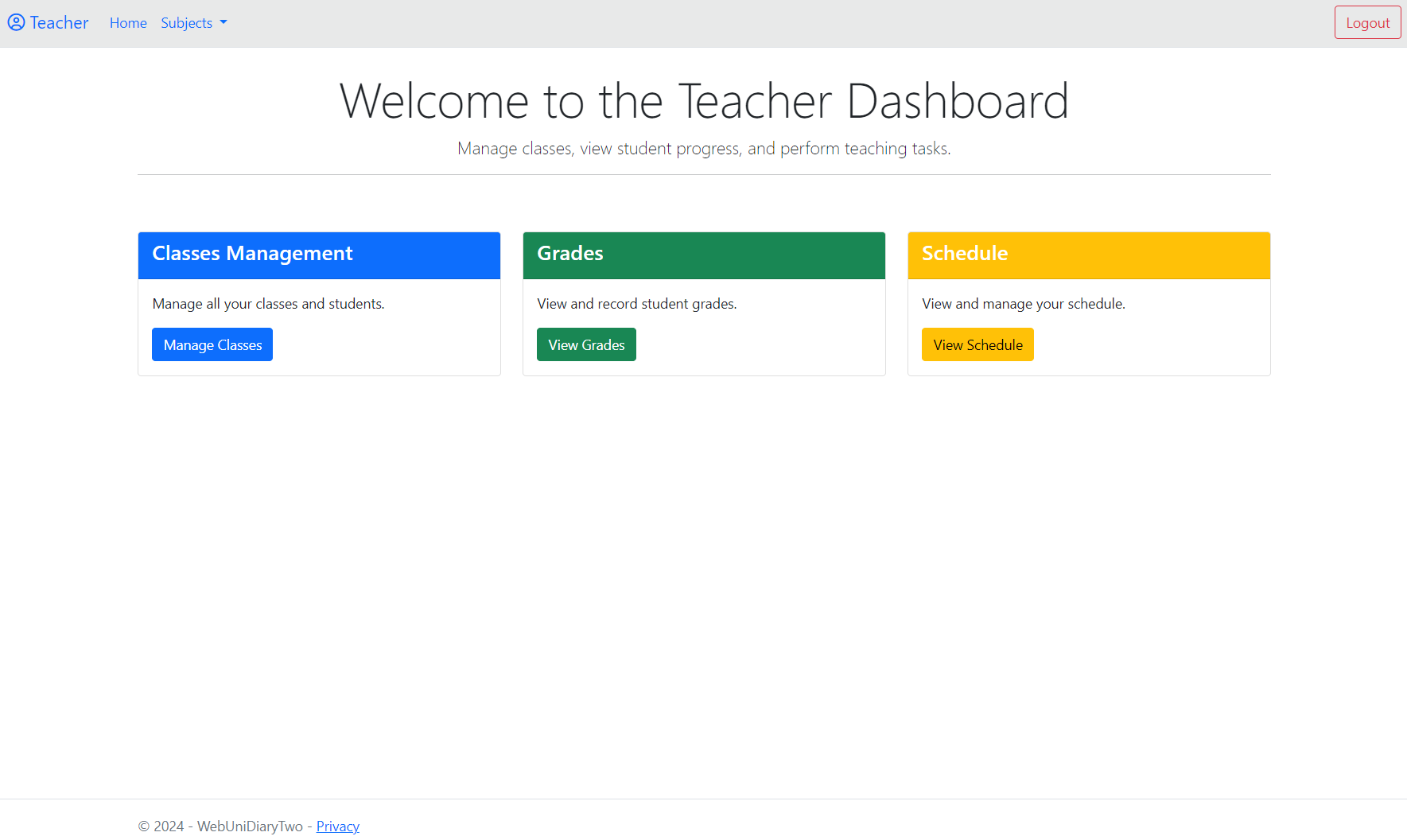


*Фигура 6.6 – SemesterOverview може да се види академичното състояние на студентът*

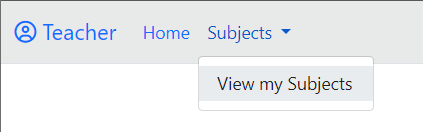
### **6.4 Употреба на приложениетж като преподавател**

След вписване в системата (Фигура 6.3) преподавателите ще бъдат пренасочени към главната страница контролера.

От тук на татък те имат достъп до всички инструменти предназначени за преподавателите.



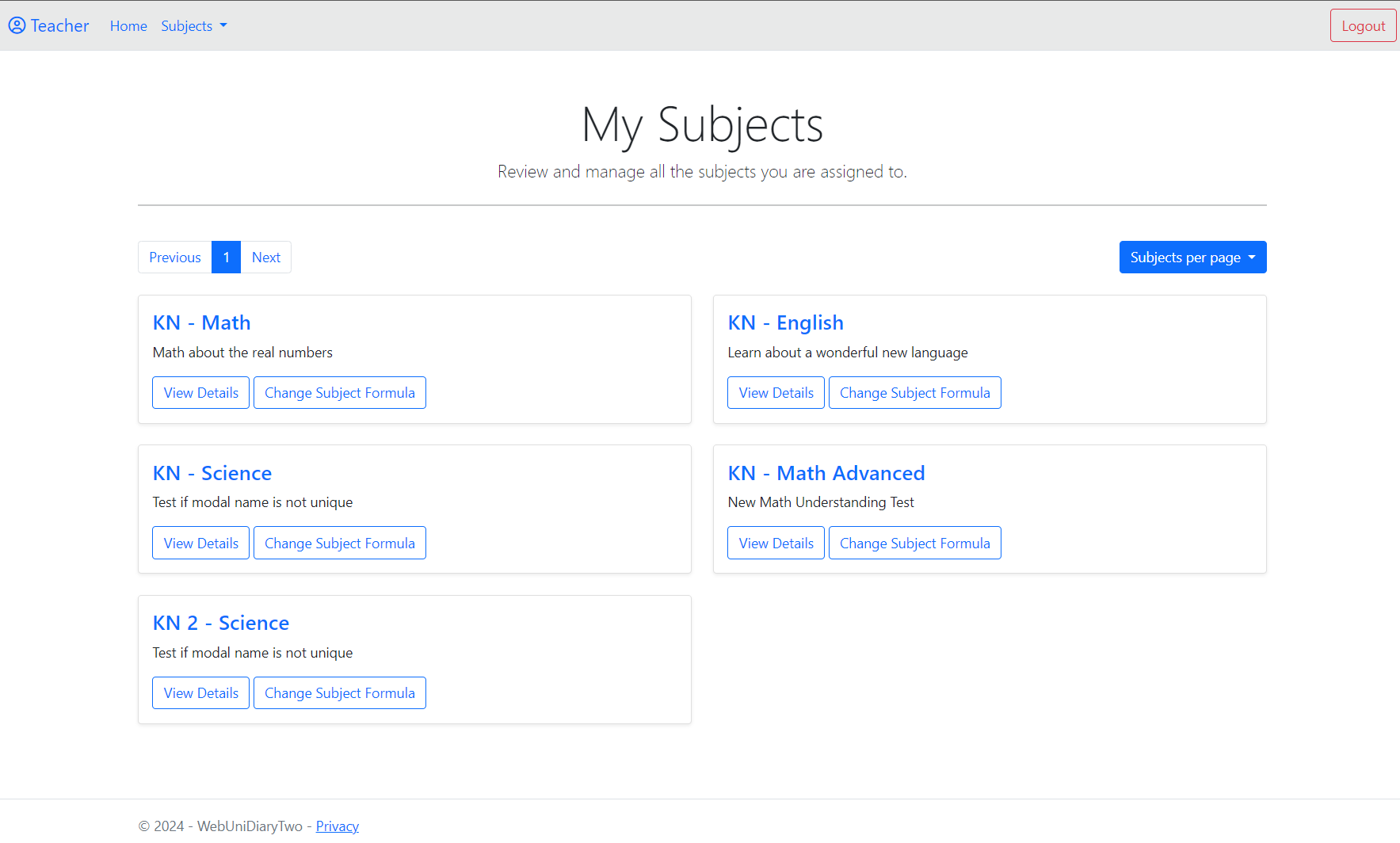
*Фигура 6.7 – Главна страница на Преподавател – Teacher/Index*



*Фигура 6.8 – Разглеждане на предметите на даден преподавател*

При избор на опцията за разглеждане на предметите (Фигура 6.8), преподавателят се пренасочва към Teacher/BrowseSubjects, откъдето има достъп до всеки предмет, по който преподава. Предметите са подредени според специалностите, които ги изучават. На Фигура 6.9 може да се види как изглежда примерна страница.

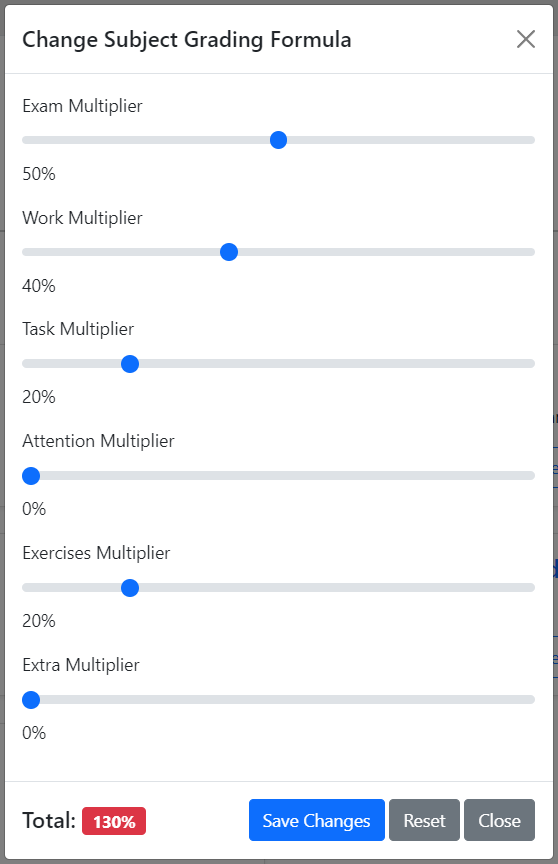
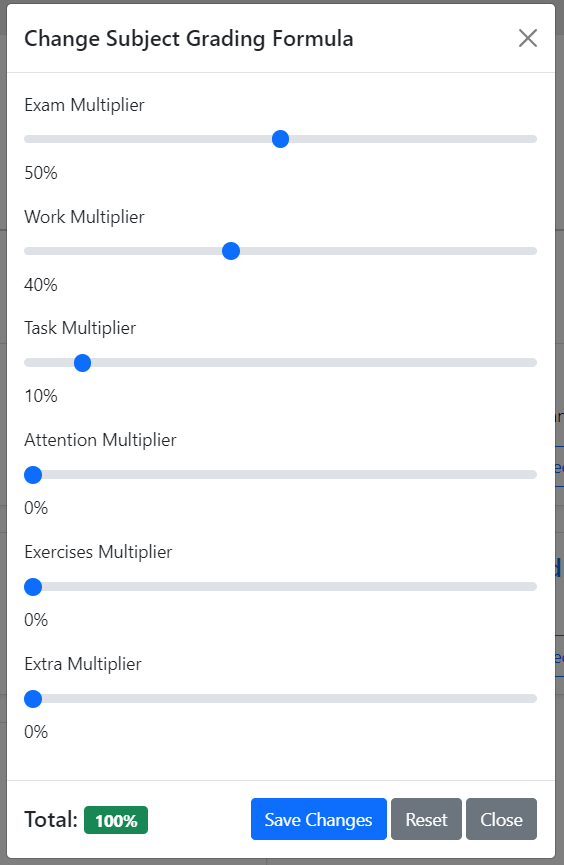
Освен възможността да прегледа всеки предмет, преподавателят може също да променя формулата, с която се изчислява успехът по дисциплината.



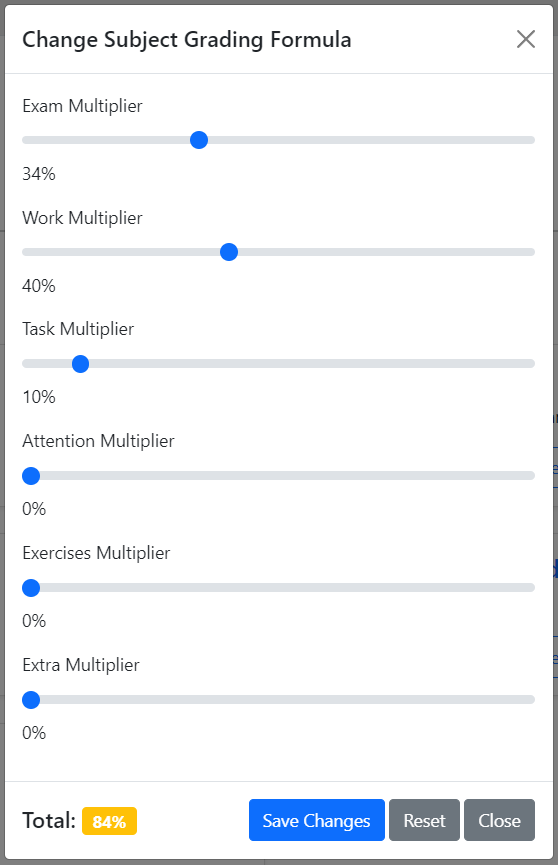
Фигура 6.9 – Страница Teacher/BrowseSubjects

При избор на Change Subject Formula се зарежда модално меню, в което се показва текущата формула, ако има такава. И потребителят има пълен достъп да променя процентите по пресмятането на крайната оценка.

В процесът по промяната на процентите има няколко възможни ситуации:

* Когато общият резултат е 100%, формулата може да се запази успешно в базата данни – пример за това е Фигура 6.10.
* Ако формулата надхвърля 100%, то тя се води за неуспешна и не може да бъде запазена. Пример за това е Фигура 6.11.
* Последен вариант, ако формулата е под 100%, то тя се води за незавършена и не може да се запази в приложението. Пример за това е Фигура 6.12

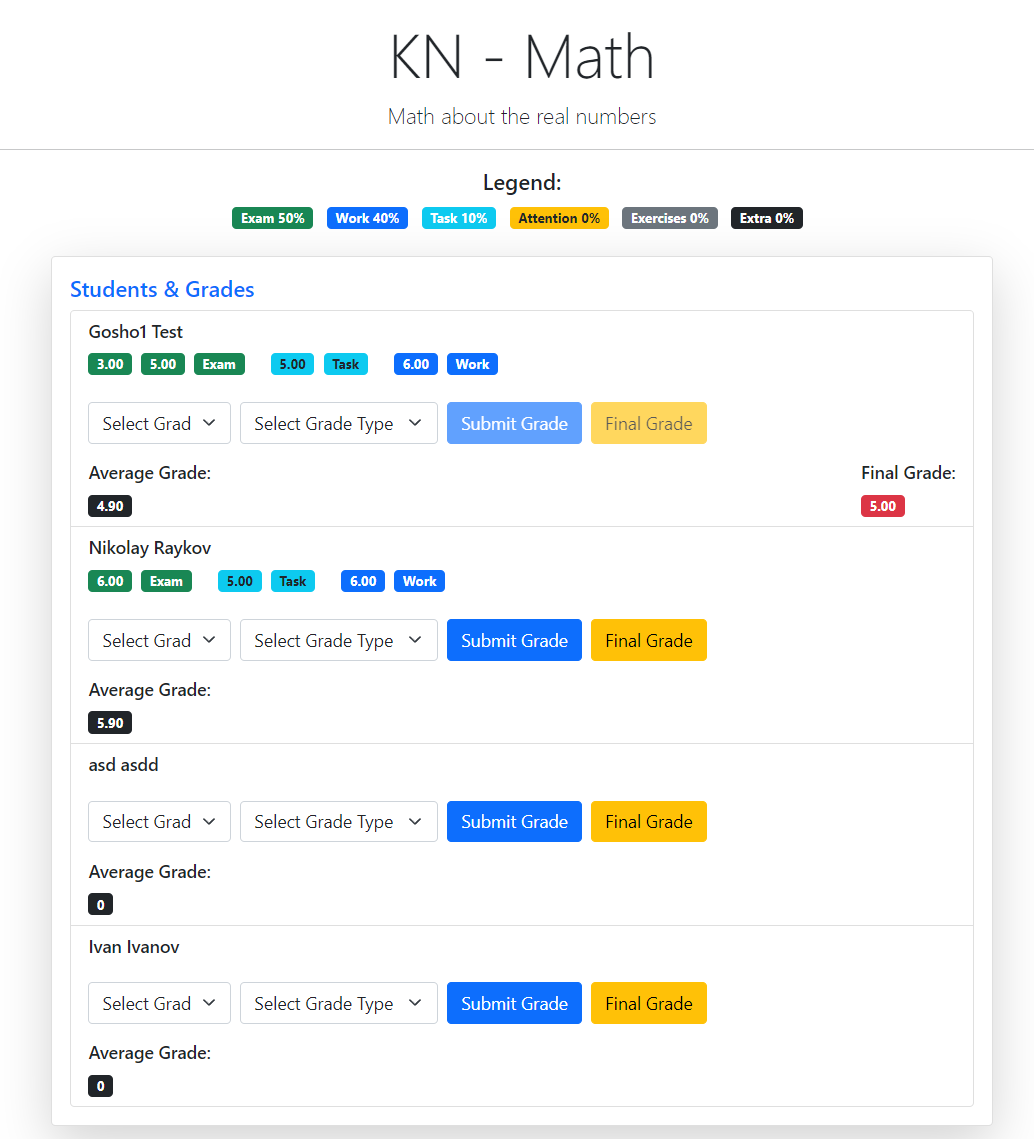
*Фигура 6.10 – Успешна формула Фигура 6.11 – Не успешна формула*

**

*Фигура 6.12 – Недовършена формула*

При избор на View Details от Фигура 6.9 за някой от предметите, преподавателят има възможността да разгледа подробните данни за този предмет и студентите от специалността. В новия изглед се предлага възможност за нанасяне на оценки и оформянето на крайната оценка.

Пример за това е Фигура 6.13, където е показан примерен дневник на студентите и техните оценки.



*Фигура 6.13 – CourseDetails – Студенти изучаващи предмет от дадена специалност*

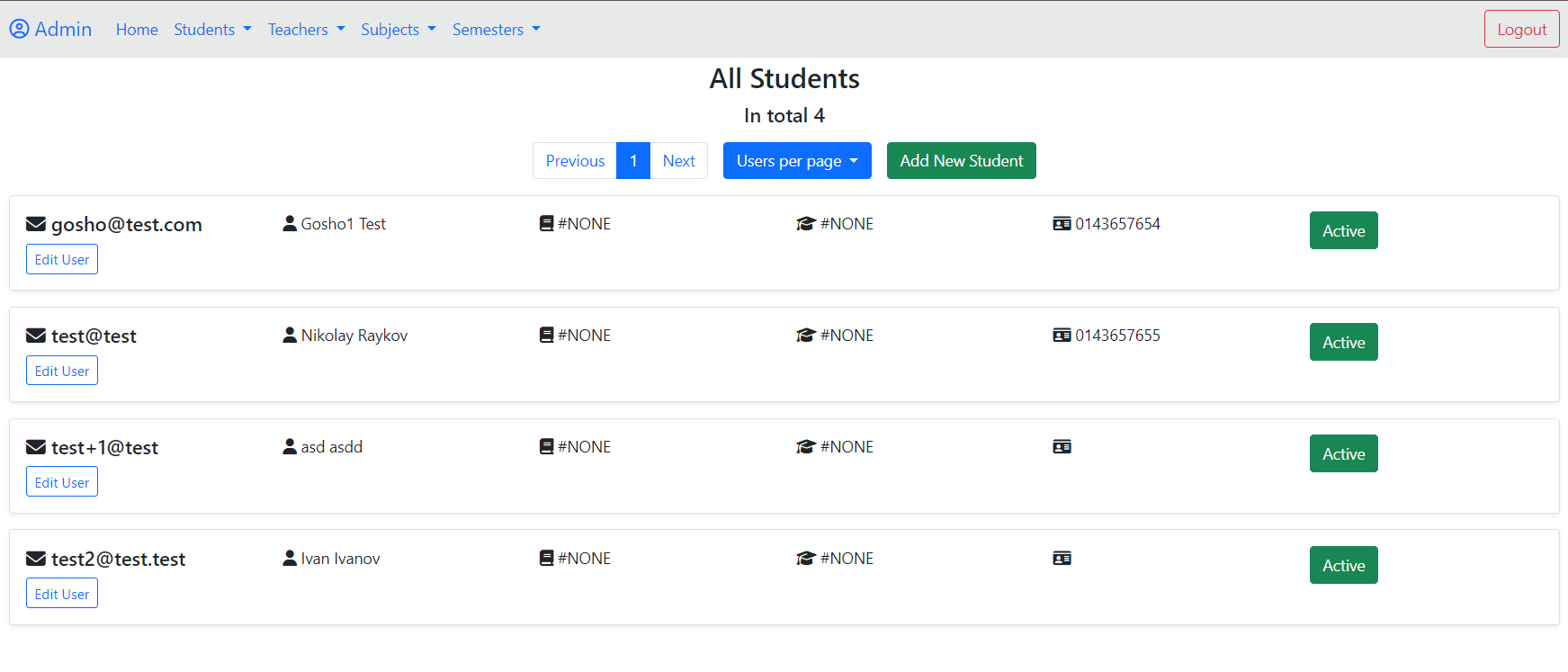
Преподавателят лесно може да избере дадана оценка и да я нанесе стига студентът да няма нанесена крайна оценка.

### **6.5 Употреба на приложението като администратор**

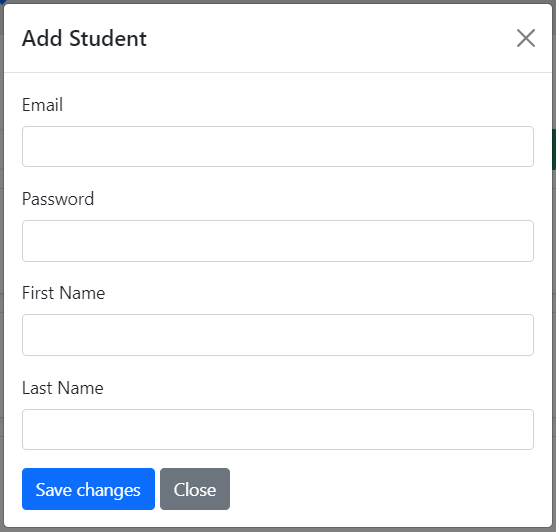
При успешно вписване в системата, администраторът ще бъде приветстван от началната страница Admin/Index. Тук са представени различни линкове за бърз достъп до различни страници.

Първо ще разгледаме страницата с всички студенти на приложението. Фигура 6.14 показва тази страница, където лесно може да се прегледа всеки студент, както и да се добави нов при нужда. Страницата също така поддържа редактиране на вече съществуващи студенти.

Добавянето и редактирането се извършват чрез модална форма на същата страница. Пример за това е показан на Фигура 6.15.

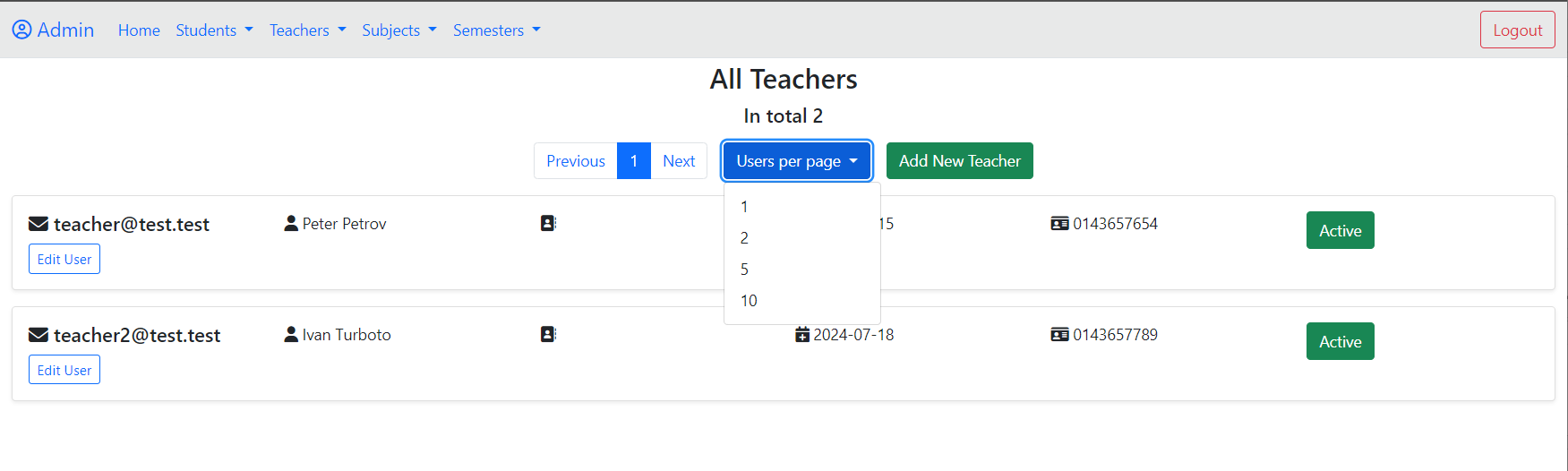


*Фигура 6.14 – Страница Admin/BrowseStudents*



*Фигура 6.15 – Модал форма за добавяне на нов студент*

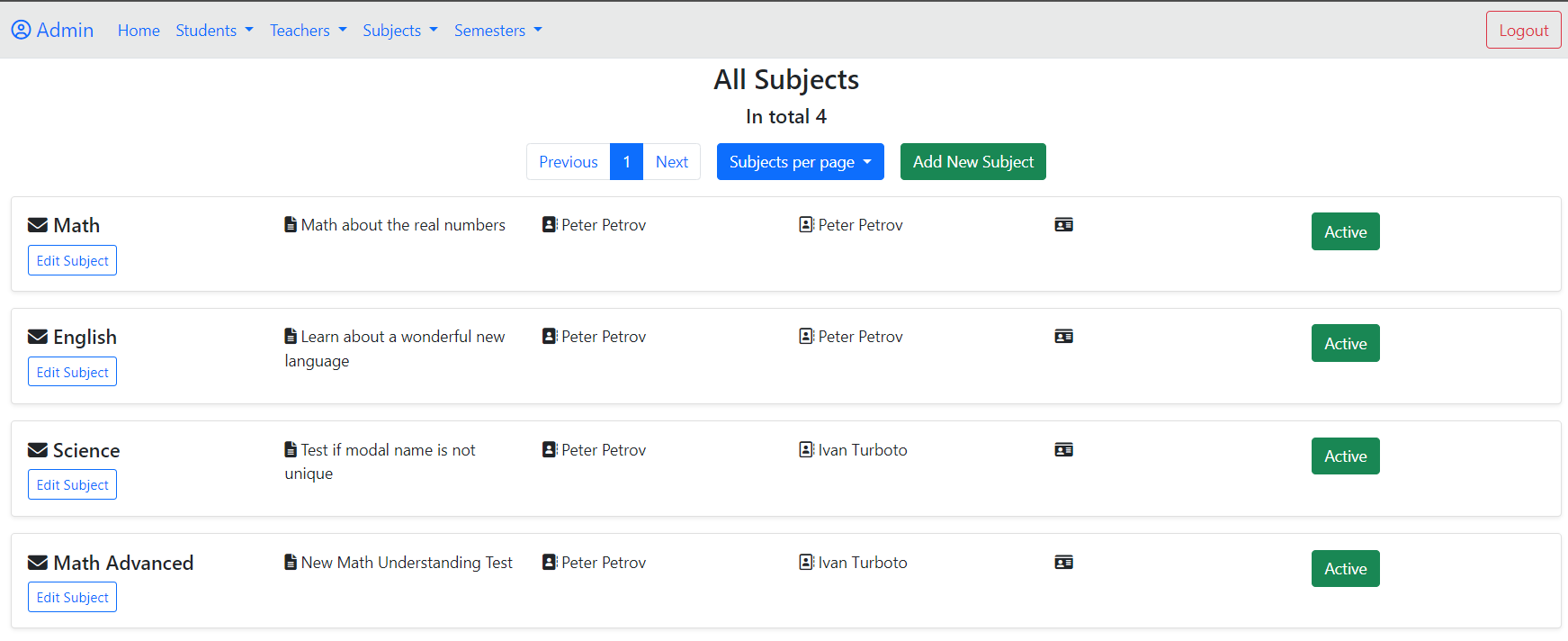
Друга страница, достъпна за администраторите, е тази с всички преподаватели. Тя се намира на **Admin/BrowseTeachers**. Тук се предлага идентична функционалност като при страницата за студентите – добавяне, редактиране и преглеждане на преподаватели.



*Фигура 6.16 – Страница Admin/BrowseTeachers*

Следващата страница е предназначена за показване на учебните предмети. Тук могат да се добавят, обновяват и преглеждат предмети. Страницата е подобна на предишните две.

По-специфичното е, че всеки предмет трябва да има преподавател, а в случай че е само един, той се добавя и като заместник. В бъдеще ще бъде добавена поддръжка за уникална икона за всеки предмет, която преподавателите ще могат да променят. В момента текущата икона е просто запълнено писмо.

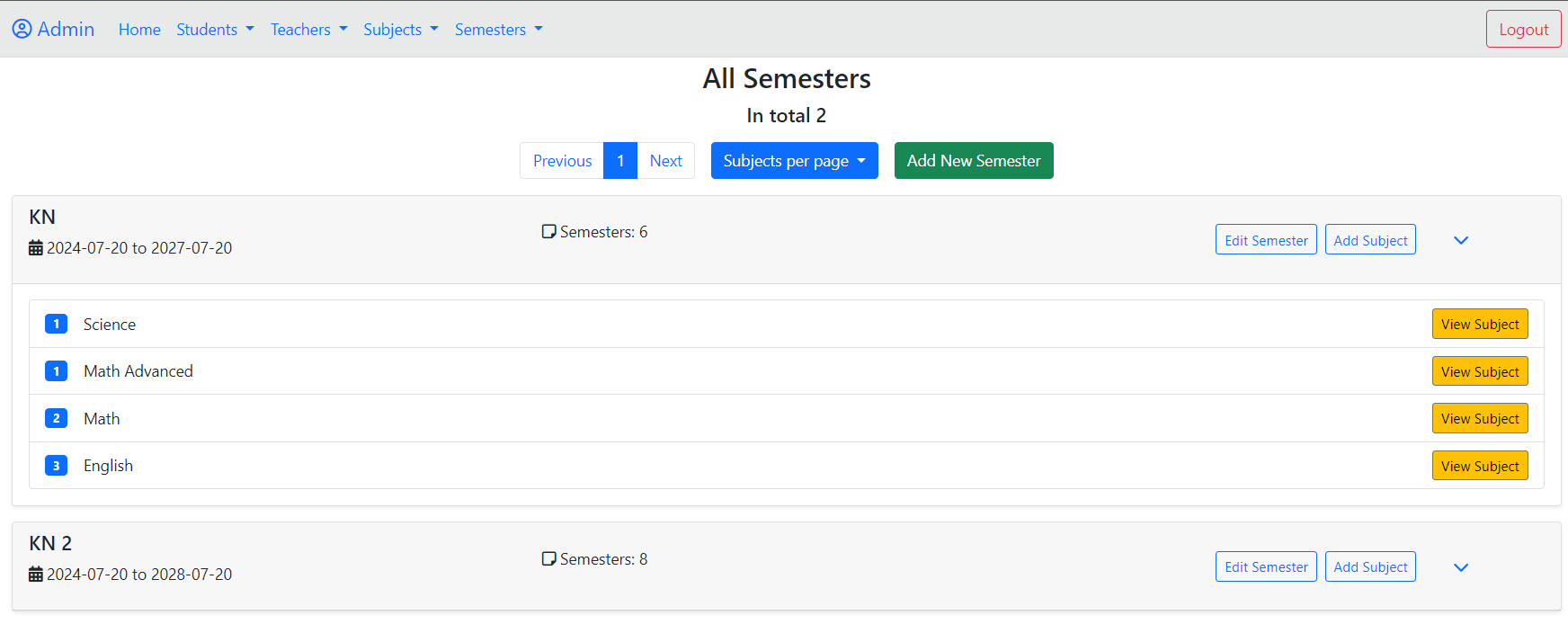


*Фигура 6.17 – Странциа Admin/BrowseSubjects*

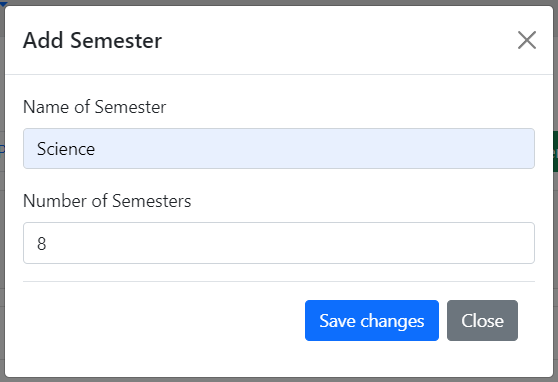
Последната страница, достъпна за администраторите на този етап, е страницата за всички специалности. Тук може да се види всяка специалност и нейният учебен план. Фигура 6.18 показва тази страница.

Дисциплините са подредени според семестъра, в който се изучват. Поддържа се промяна на данните за специалността. Добавянето на нова специалност е показано на Фигура 6.19, както и добавянето на нов предмет към дадена специалност чрез бутона Add Subject. Пример за тази функционалност е показан на Фигура 6.20.

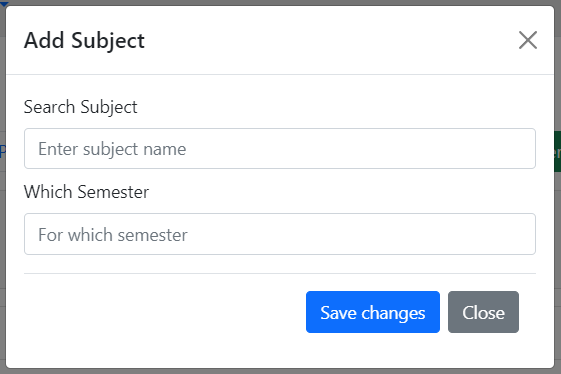
Няколко важни стъпки: специалностите трябва да имат четен брой семестри, в противен случай не могат да бъдат запазени в системата. Действително, не е възможно да има учебна година, състояща се от само един семестър.



*Фигура 6.18 – Страница Admin/BrowseSemesters*

**

*Фигура 6.19 – Модална форма за добавяне на нова Специалност*

**

*Фигура 6.20 - Модално меню за добавяне на нов предмет*

При избор на бутон View Subject от Фигура 6.18 се отваря отделна страница, къде се показват всички студенти и техният текущ успех. В момента не се поддържа изтриване на оценките, но вбъдеще ще бъде достъпно чрез натискане на желаната оценка.

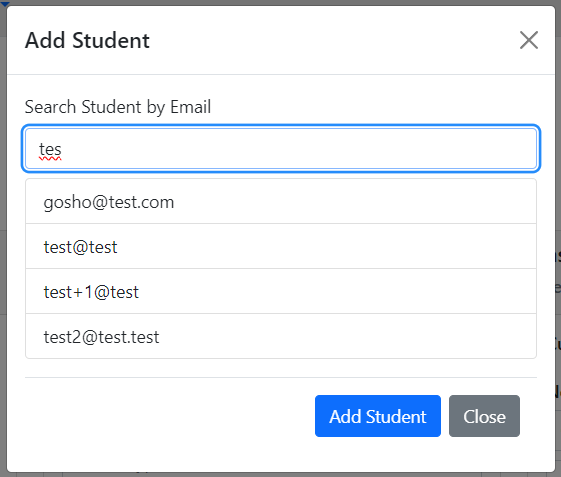
Тук освен да се разглеждат оценките на различните студенти, могат и да се добавят нови, при случай че липсват или е настъпила грешка при преподавателя.

Тази страница Admin/BrowseGrades може да се види на Фигура 6.21, друга важна функционалност е добавянето на нов студент да изучава дадена специалност. Това се извършва от бутон Add Student. И може да се прави по всяко време.

Пример за това може да се разгледа на Фигура 6.22. Тук чрез ajax заявка се предлагат студенти отговарящи на текстовият критерии. При намиране на правлният нов студент, администратора трябва единствено да натисне потребителя, който иска да добави.



*Фигура 6.21 – Всички студенти от дадена специалност и техният успех по даден предмет*



Фигура 6.22 – Добавяне на нов студент, чрез търсене по имейл

# **7. Основни резултати, изводи и препоръки**

Основният резултат е създаването на приложение, пригодено както за настолни компютри, така и за мобилни устройства, което позволява съхранението на учебни данни – било то за потребители или учебни предмети. Това е постигнато чрез използването на модерни технологии и утвърдени принципи в разработката.

Проектът улеснява модификацията на данните и ефективното разрешаване на проблемите, свързани с оценяването на студентите, като предлага ясен и лесен модел за изчисляване на примерни крайни оценки.

Системата позволява редактирането на множество данни, без риск от загуба на предишна информация, като същевременно гарантира гъвкавост и устойчивост при бъдещи актуализации. Приложението е проектирано така, че да бъде лесно за поддръжка и надграждане, което го прави подходящо за продължително използване и разширяване в образователната сфера.

# **8. Използвана литература**

|  |  |
| --- | --- |
| [1] | B. Atanasov и K. Shoilekova, Online Calculator for Determining the Optimal Seeding, Ruse: UNIVERSITY OF RUSE “Angel Kanchev”, 2023. |
| [2] | D. Abramov и A. Clark, „react-redux,“ [Онлайн]. Available: https://react-redux.js.org/. [Отваряно на 18 3 2025]. |
| [3] | netcup GmbH, „vsp,“ [Онлайн]. Available: https://www.netcup.com/en/server/vps. [Отваряно на 18 3 2025]. |
| [4] | Microsoft, „visualstudio,“ Microsoft, [Онлайн]. Available: https://code.visualstudio.com/. [Отваряно на 17 3 2025]. |
| [5] | Microsoft, „typescriptlang,“ Microsoft, [Онлайн]. Available: https://www.typescriptlang.org/. [Отваряно на 17 3 2025]. |
| [6] | Cope, „tgw-calculator,“ [Онлайн]. Available: https://copeseeds.co.uk/services/tgw-calculator/. [Отваряно на 17 3 2025]. |
| [7] | Tailwind Labs Inc., „tailwindcss,“ [Онлайн]. Available: https://tailwindcss.com/. [Отваряно на 18 3 2025]. |
| [8] | Agrii, „seed-rate-calculator,“ Agrii, [Онлайн]. Available: https://www.agrii.co.uk/calculators/seed-rate-calculator/. [Отваряно на 17 3 2025]. |
| [9] | preparati.info, „raztvor,“ [Онлайн]. Available: https://preparati.info/calculator/raztvor. [Отваряно на 17 3 2025]. |
| [10] | Prisma Data, Inc., „Prisma,“ Prisma Data, Inc., [Онлайн]. Available: https://www.prisma.io/. [Отваряно на 17 3 2025]. |
| [11] | The PostgreSQL Global Development Group, „postgresql,“ The PostgreSQL Global Development Group, [Онлайн]. Available: https://www.postgresql.org/. [Отваряно на 17 3 2025]. |
| [12] | Реколта.bg, „planove,“ Реколта digital, [Онлайн]. Available: https://digital.rekolta.bg/planove/. [Отваряно на 17 3 2025]. |
| [13] | Vercel, „nextjs.org,“ Vercel, [Онлайн]. Available: https://nextjs.org/. [Отваряно на 17 3 2025]. |
| [14] | OpenJS Foundation, „jestjs,“ [Онлайн]. Available: https://jestjs.io/. [Отваряно на 18 3 2025]. |
| [15] | Microsoft, „github,“ [Онлайн]. Available: https://github.com/. [Отваряно на 18 3 2025]. |
| [16] | Docker, Inc., „Docker,“ Docker, Inc., [Онлайн]. Available: https://www.docker.com/. [Отваряно на 17 3 2025]. |
| [17] | Cloudflare, Inc., „cloudflare,“ [Онлайн]. Available: https://www.cloudflare.com/. [Отваряно на 18 3 2025]. |
| [18] | Microsoft, „actions,“ [Онлайн]. Available: https://github.com/features/actions. [Отваряно на 18 3 2025]. |
| [19] | agrocalculator.bg, „/калкулатори/калкулатор-за-препарати/хербициди/,“ [Онлайн]. Available: https://www.agrocalculator.bg/%d0%ba%d0%b0%d0%bb%d0%ba%d1%83%d0%bb%d0%b0%d1%82%d0%be%d1%80%d0%b8/%d0%ba%d0%b0%d0%bb%d0%ba%d1%83%d0%bb%d0%b0%d1%82%d0%be%d1%80-%d0%b7%d0%b0-%d0%bf%d1%80%d0%b5%d0%bf%d0%b0%d1%80%d0%b0%d1%82%d0%b8/%d1%85%d0%b5%d1%80%d0%b1%d0%b. [Отваряно на 17 3 2025]. |
| [20] | A. Williams, „GitHub Pours Energies into Enterprise – Raises $100 Million From Power VC Andreessen Horowitz,“ TechCrunch, [Онлайн]. Available: https://techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/. [Отваряно на 18 3 2025]. |
| [21] | L. Torvalds, „git-scm,“ Microsoft, [Онлайн]. Available: https://git-scm.com/. [Отваряно на 17 3 2025]. |
| [22] | shadcn, „shadcn,“ [Онлайн]. Available: https://ui.shadcn.com/. [Отваряно на 18 3 2025]. |
| [23] | C. McDonnell, „zod,“ [Онлайн]. Available: https://zod.dev/. [Отваряно на 18 3 2025]. |
| [24] | E. Fennis, K. Rigó и J. Guddas, „icons,“ [Онлайн]. Available: https://lucide.dev/. [Отваряно на 18 3 2025]. |

# **9. Таблица с фигури**

[Фигура 1 Cope 6](#_Toc200639630)

[Фигура 2. Реколта digital 7](#_Toc200639631)

[Фигура 3 Agrii 8](#_Toc200639632)

[Фигура 4 agrocalculator.bg 9](#_Toc200639633)

[Фигура 5 preparati.info 10](#_Toc200639634)

[Фигура 6 Диаграма на случаите на употреба 18](#_Toc200639635)

[Фигура 7 Диаграма на дейностите – създаване на нова специалност 19](#_Toc200639636)

[Фигура 8. Релационен модел на базата данни 20](#_Toc200639637)

[Фигура 9 A - Файлова структура 26](#_Toc200639638)

[Фигура 9 B - Файлова структура 26](#_Toc200639639)

[Фигура 9 C - Файлова структура 27](#_Toc200639640)

[Фигура 9 D - Файлова структура 27](#_Toc200639641)

# **10. Таблица с таблици**

[Таблица 1 Сравнение на функционалности 11](#_Toc200639556)

[1] Какво представлява Школо, URL: <https://bg.wikipedia.org/wiki/%D0%A8%D0%BA%D0%BE%D0%BB%D0%BE>, посетен на 02.02.2024г.

### [2] AdminPlus, URL: https://adminplus.bg/, посетен на 07.02.2024г.

### [3] Administrator's Plus, URL: <https://www.rediker.com/>, посетен на 07.02.2024г.

### [4] ASP.NET Core, URL https://dotnet.microsoft.com/en-us/apps/aspnet, посетен на 10.02.2024г.

### [5] Microsoft SQL Server, URL: https://www.microsoft.com/en-us/sql-server/sql-server-downloads, посетен на 10.06.2024г.

### [6] Entity Framework Documentation Hub, URL: https://learn.microsoft.com/en-us/ef, посетен на 10.06.2024г.

### [7] Какво представлява унифицирания език за моделиране (UML), URL: <https://www.microsoft.com/bg-bg/microsoft-365/visio/uml>, посетен на 20.07.2024г.