**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 (Atanasov & Shoilekova, 2023).

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**

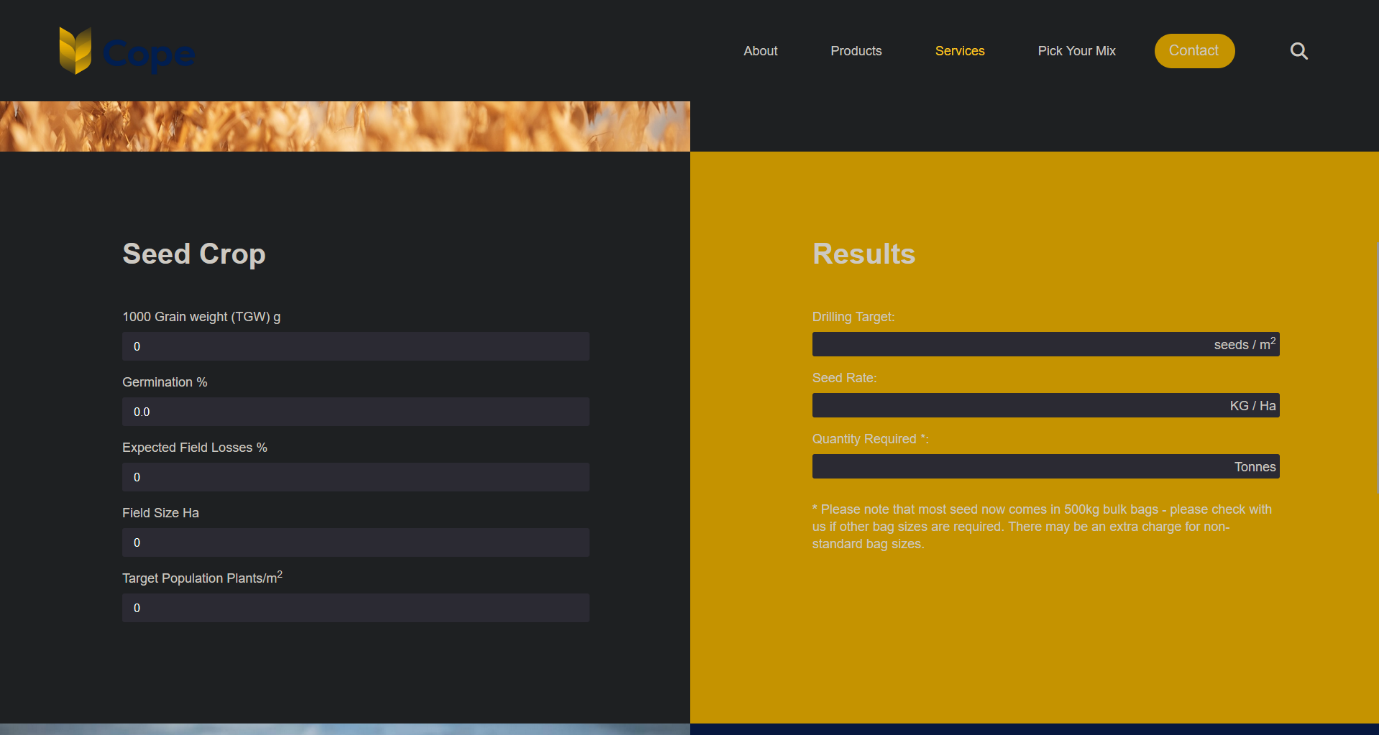
Страница направена от английска фирма занимаваща се с продажбата на семена и торове (Фигура 1) (Cope, n.d.). Сайтът е главно насочен към продажби, като допълнение има базов калкулатор, който не разполага с научно доказани данни. Продукта може да се види на.

**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 1 Cope

### **2.1.2 Реколта digital**

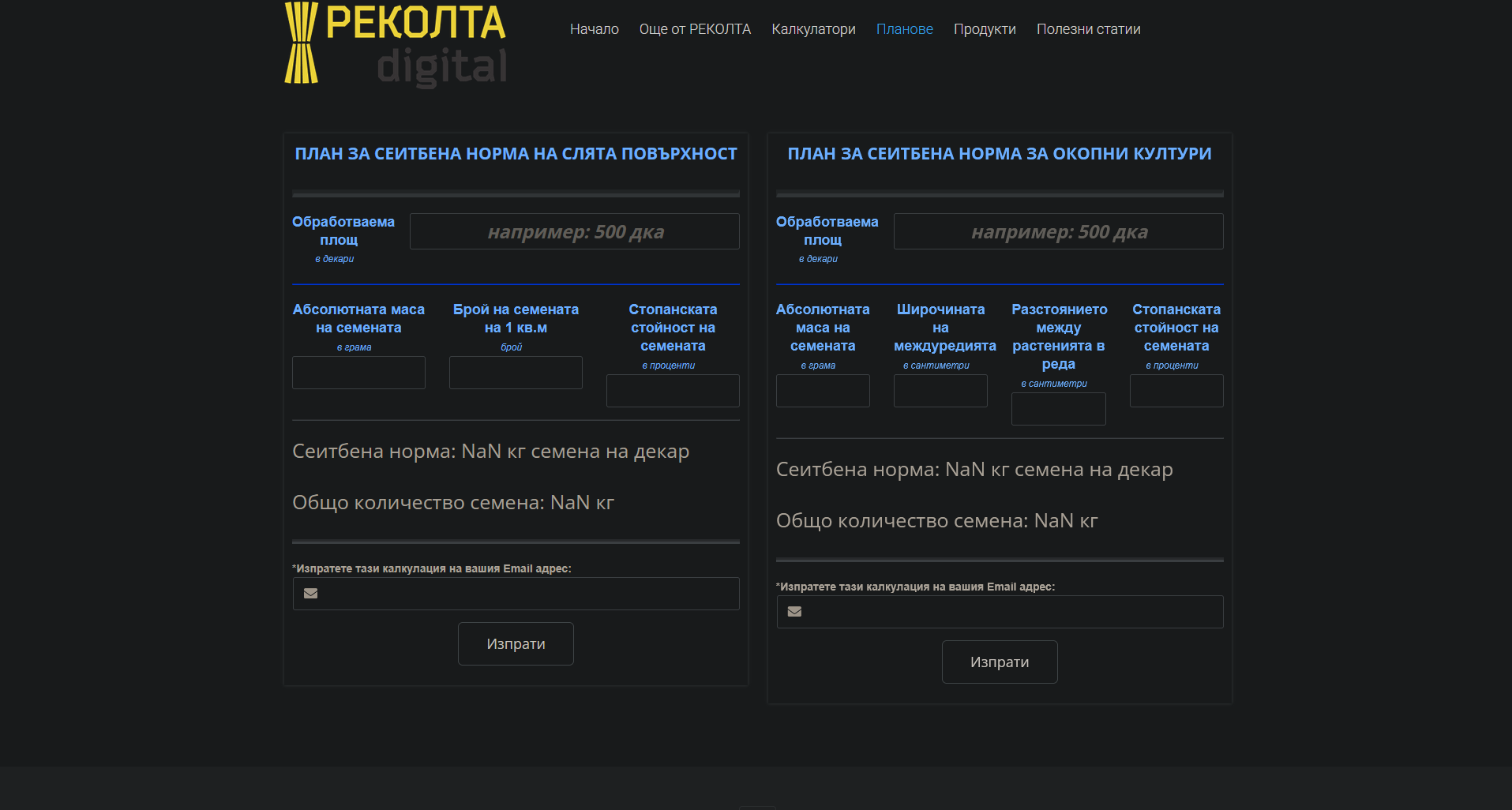
Подобно на Cope, Реколта digital също предлага подобен калкулатор за изчисляване на сеитбена норма за основни култури като пшеница и рапица засягащи всякакви черти от отглеждането на тези култури (Реколта.bg, n.d.).

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 2. Реколта digital

### **2.1.3 Agrii**

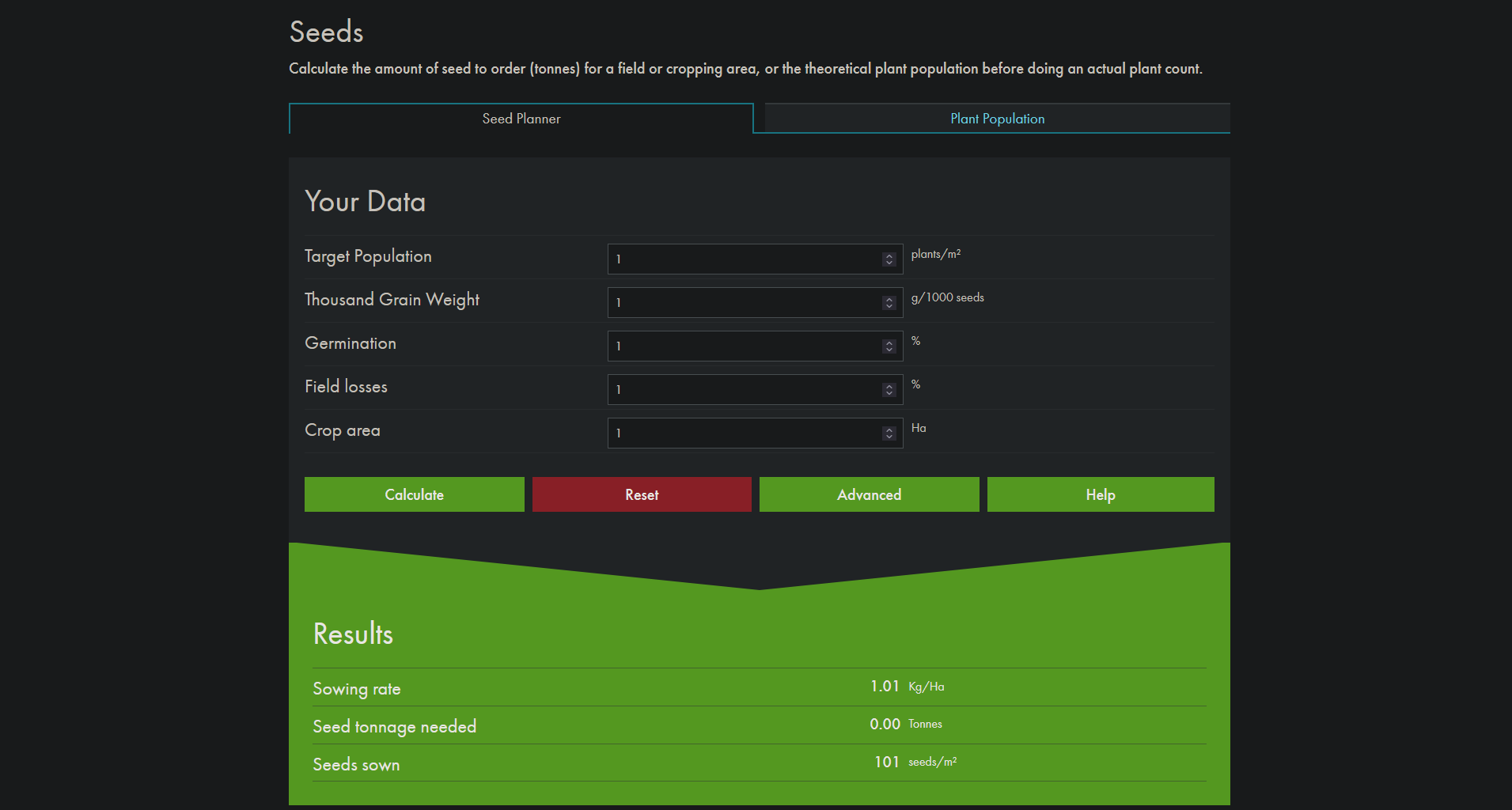
Agrii, подобна на Cope, е английска фирма занимаваща се с консултация и продажба на семена и торове. Основната цел на сайтът е продажба, а калкулаторът е базов като при Cope (Agrii, n.d.).

**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 3 Agrii

### **2.1.4 Агро Калкулатор**

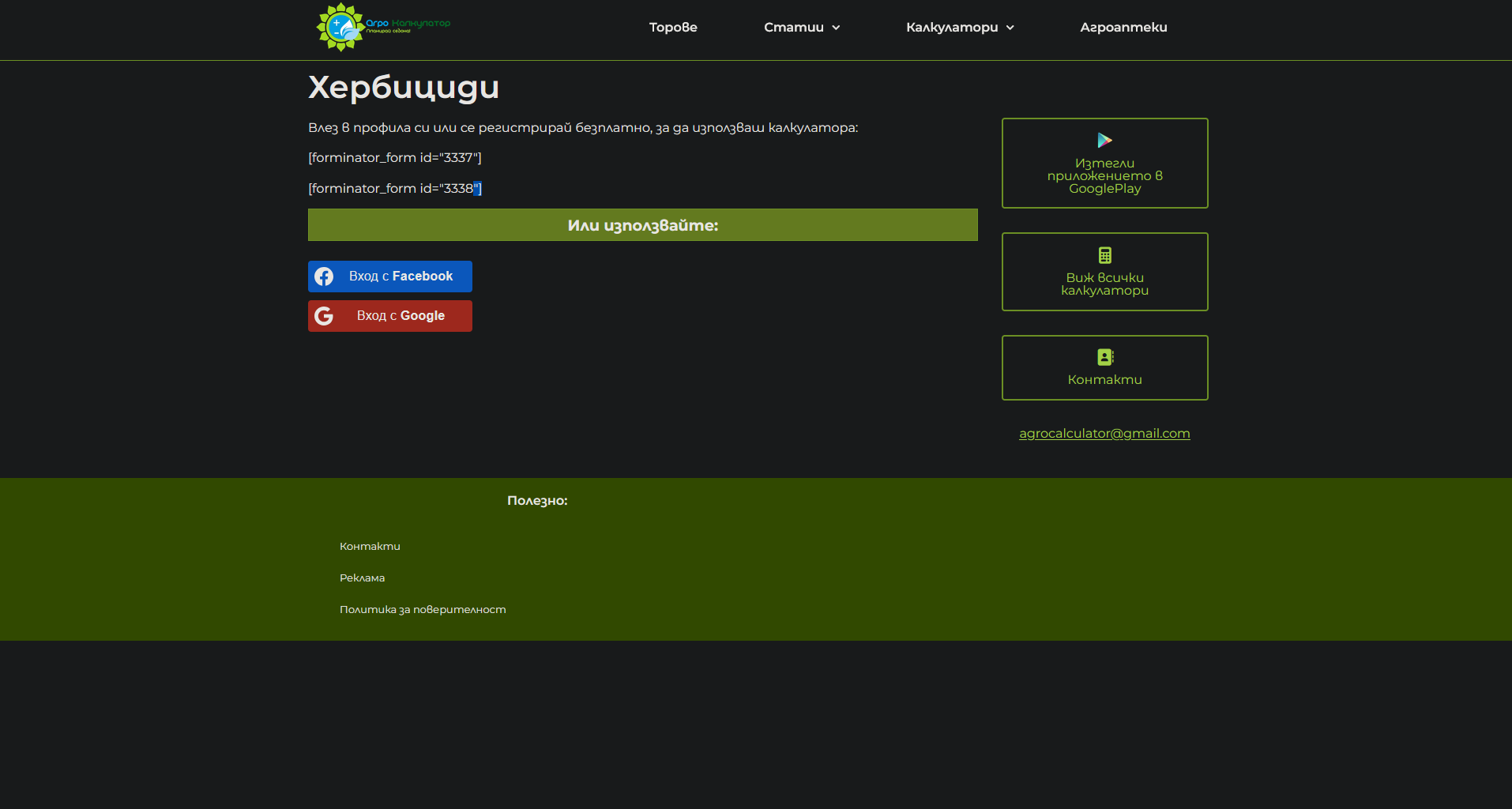
Agrocalculator.bg е български сайт, съвкупност от няколко калкулатора със земеделски цели. За съжаление не работи (agrocalculator.bg, n.d.).

**Предимства:**

* има мобилно приложение.

**Недостатъци:**

* сайтът и калкулаторите не работят.



Фигура 4 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 (preparati.info, n.d.).

**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 5 preparati.info

## **2.2 Анализ на съществуващите системи**

Сравнение на функционалните възможности на различните продукти в таблица 1.

Таблица 1 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) (Microsoft, n.d.).

#### **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 (Torvalds, n.d.).

#### **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 (Docker, Inc., n.d.).

## **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 (Meta Platforms, Inc., n.d.).

## **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 (Vercel, n.d.).

## **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 (Prisma Data, Inc., n.d.).

## **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 (The PostgreSQL Global Development Group, n.d.).

#### **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 предвиди мост и стабилност при разработката на сложни приложения (Microsoft, н.д.).

#### **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 (Abramov & Clark, н.д.).

#### **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 (McDonnell, н.д.).

#### **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 (OpenJS Foundation, н.д.).

#### **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 (shadcn, н.д.).

#### **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 (Tailwind Labs Inc., n.d.).

#### **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 (Fennis, Rigó, & Guddas, н.д.).

#### **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 (Williams, н.д.) (Microsoft, н.д.).

#### **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 (Microsoft, н.д.).

#### **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 (netcup GmbH, н.д.).

#### **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 (Cloudflare, Inc., н.д.).

## **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**

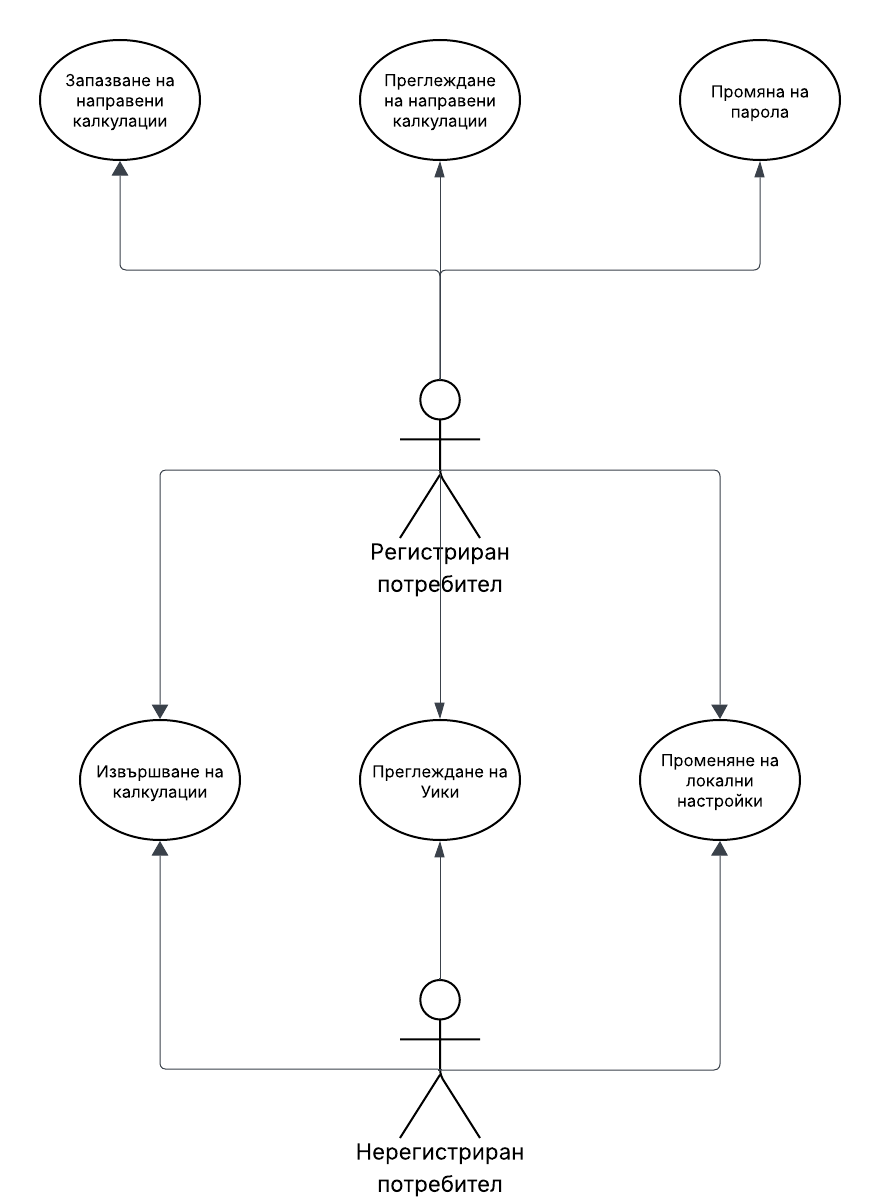
Това е една от най-използваните диаграми, тъй като визуално представя функционалността на системата от гледната точка на потребителя. Главната идея е да се демонстрира взаимодействието на потребителите със системата. Действията се наричат случаи на употреба, а хората или другите взаимодействащи системи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

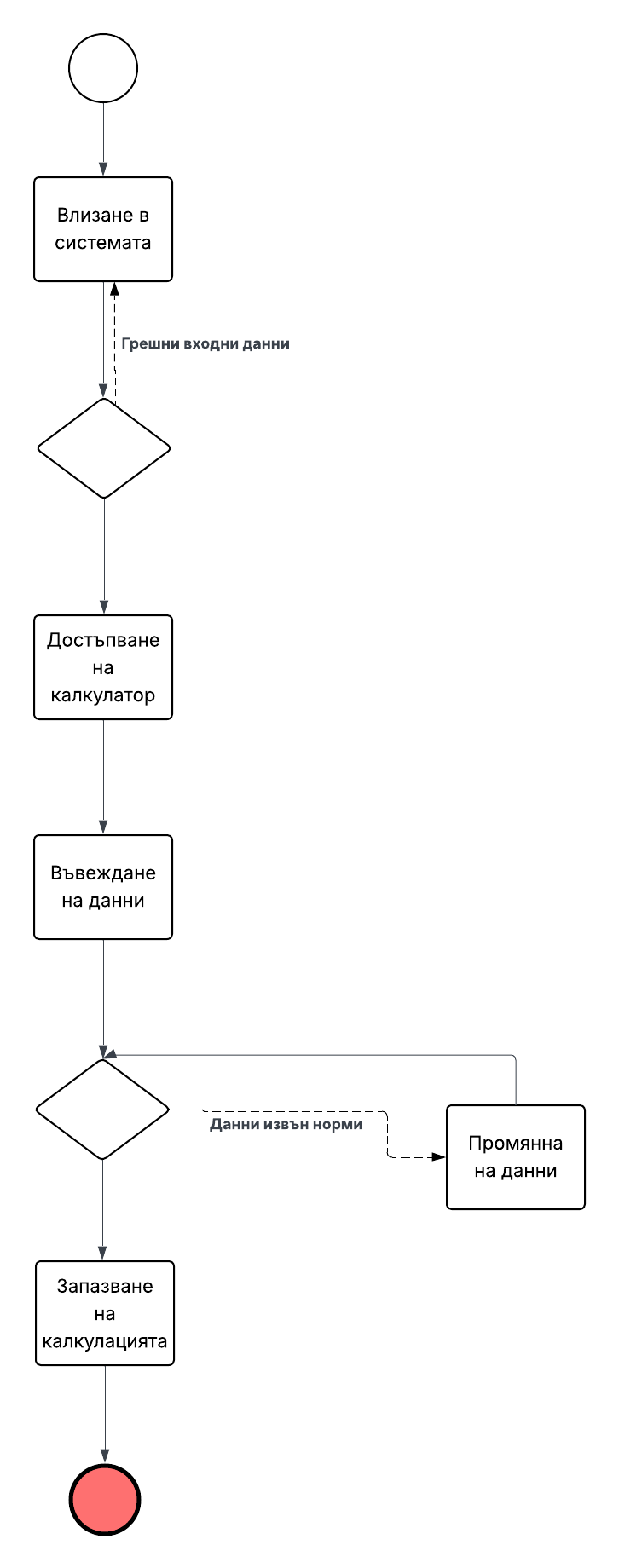
**Регистриран потребител** – вече създал профил и автентикиран в системата. Има разширен достъп до функционалностите, включително всички налични за нерегистриран потребител. Освен това, той може да запазва направените от него калкулации, да ги преглежда по-късно, както и да променя своята парола. Също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



Фигура 6 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 7 Диаграма на дейностите – създаване на нова специалност

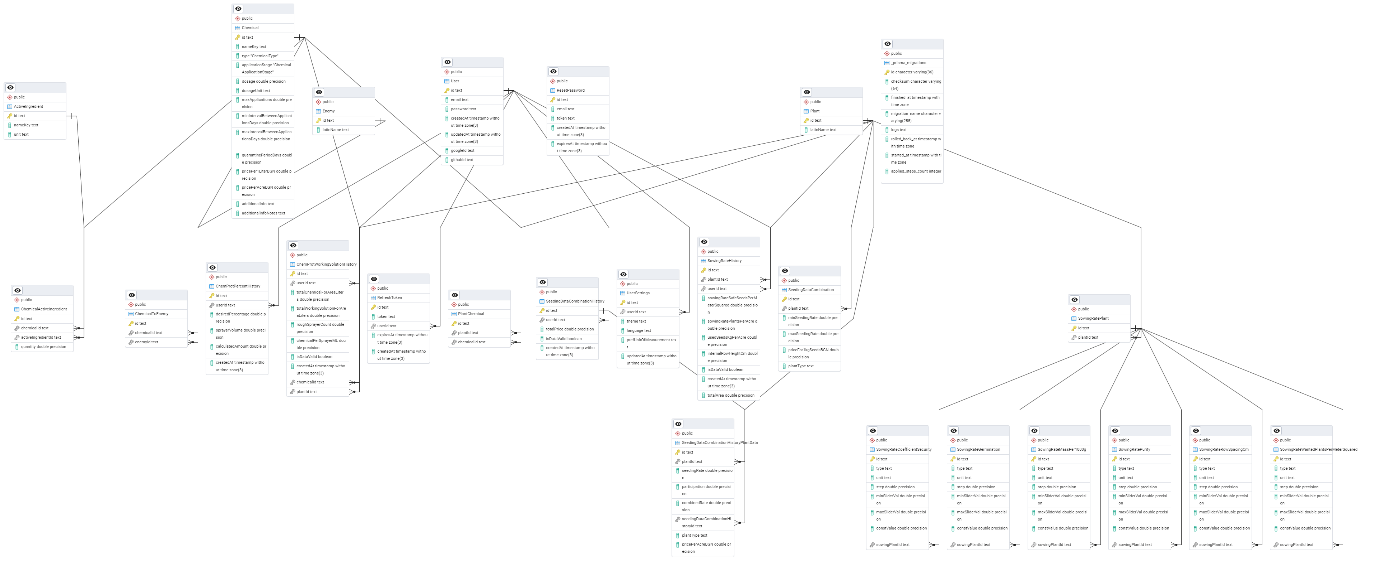
Процесът представен на Фигура 7. се отнася за основната функционалност –извършване на калкулации и тяхното запазване.

# **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 8. 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 9-12. 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 - Файлова структура - A | Фигура 10 Файлова структура - B |

|  |  |
| --- | --- |
| Фигура 11 - Файлова структура C | Фигура 12 - Файлова структура D |

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 10, 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.

A screenshot of a computer

AI-generated content may be incorrect.

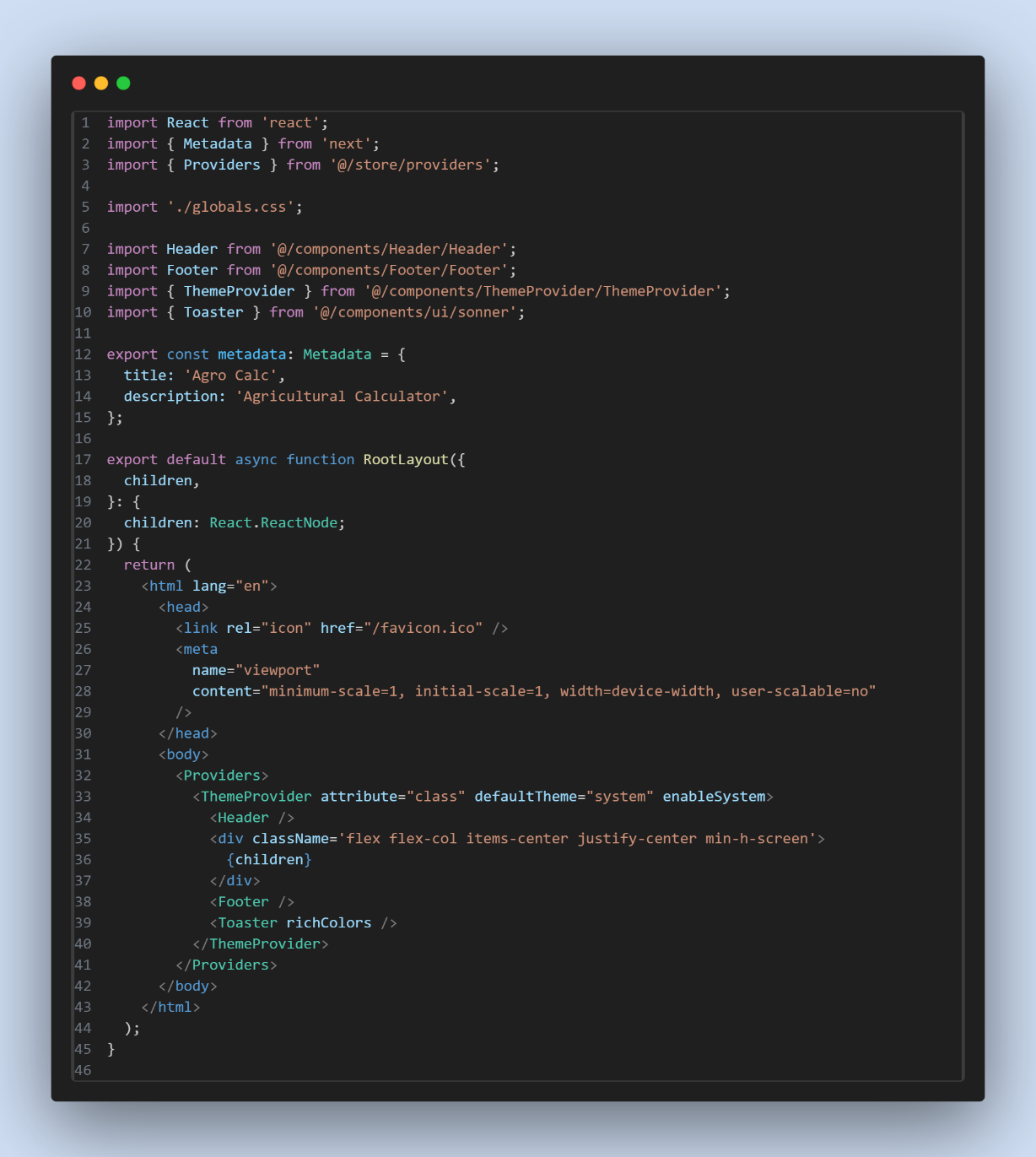
Фигура 13 - Папката /hooks

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 14. 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.



Фигура 14 - Файл layout.tsx

## **5.2 Важни файлове**

Също въпреки че всеки файл е важен за работата на проекта има няколко основни без който не може или са основна част от живота на приложението. Добър пример за такъв файл е instrumentation.tsx, той се изпълнява още в началото, след написване на старт командата и се грижи за две главни неща. Да провери дали всички environment variables (променливи от средата са инициализирани) като връзката за база от данни. Също изпълнява и много друга важна задача, при липса на данни в базата данни да въведете такива данни. В него са начални данни за базата и чрез помощта на prisma те биват вкарани.



Фигура 15 - instrumentation.tsx

Друг важен файл е middleware.tsx. Този проект не използва външни библиотеки за менажиране на автентикация. За да работи системата с акаунти и потребители който са влезли и не трябва всеки request (заявка) да се проверява от server. Логиката за това е разделена на няколко части. Първо се дефиниат всички защитени пътища. След това се взимат cookies (бисквитки) от заявката и искания път. Бисквитките са два JWT токена. JWT токен е криптографно, с алгоритъм HS256, сигурна информация за потребител. Ако изобщо не е защитен, като например началната страница, директно се пуска заявката. Ако е защитен първо се проверява дали заявката има access token (токен за достъп), това е кратко живящ токен който позволява достъп на потребителя. Ако няма директно се изпраща на страница за влизане, ако има се проверява дали е валиден. Понеже е JWT токен той може да бъде създаден и проверен с една и съща тайна фраза дефинирана в environment variables, също така може да бъде зададено и времетраене след което става не валиден. Ако не е валиден се опитва чрез refresh token (токен за презареждане) да се създаде нов access token. При провал се изпраща на страница за влизане. Ако този алгоритъм е успешен заявката се подава на компонента или API който трябва да я обработи. Това може да се види на Фигура 16.



Фигура 16 - middleware.ts

Друг важен файл отговарящ за менажирането на локланото състояние на приложението е store.ts, той може да се види на Фигура 17. За да може да се запази състоянието дори през презареждане на страницата се използва redux-persist който запазва състоянието в браузъра. След това са reducers това е начина с който redux работи с данни. За да няма обект който е с променящи се свойства има reducer и неговите actions съдържани от slice. Всеки reducer дефинира обект с променливи и за да бъде извършена промяна върху тези обекти се използва action. Всеки action взима обект подаден от програмиста и променя по строго дефиниран начин променливите от обект. Това също дава бонуса че издава сигнал който после може да бъде хванат в store.ts. Точно това се прави промяна на локалните настроики. Всеки потребител има настроики като език и тема и те биват запазени в базата данни. Като се използват тези сигнали може да се засече че има промяна и тя да бъде качена в базата, може да се засече и влизане на потребител и неговите настройки от база да бъдат смъкнати и приложени.



Фигура 17 - store.ts

## **5.3 Основни страници**

### **5.3.1 Начална страница (Index)**

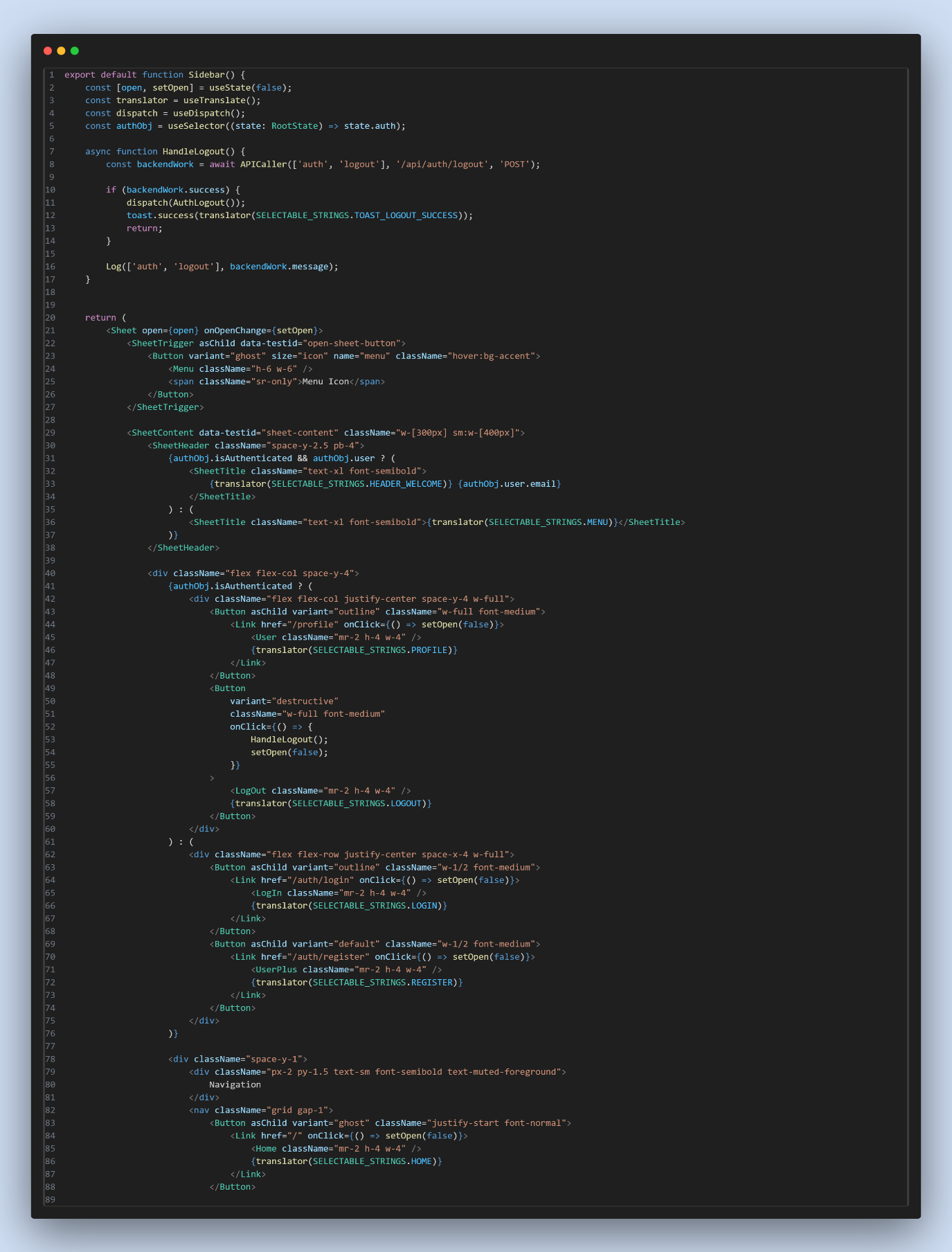
Използва се за показване на началната страница, като показва няколко снимки, обяснява функционалности и подтиква потребителя да използва калкулаторите.



Фигура 18 - Как точно изглежда една типична .tsx страница

### **5.3.2 Компонент Sidebar**

За да има достъп потребителя навсякъде достъп до бързи линкове е поставен Sidebar в Header които се намира в layout и е достъпен на всяка страница. Там отново се намират линкове към калкулатори но и към компонента за променяне на настройки като език и линкове към login и register. Също така в зависимост от текущото състояние на auth ще покажа линкове за login и register или емайл на потребителя с линкове към профил и бутон за излизане.



Фигура 19 - Обсъдения код на Sidebar

Всяка .tsx страница има 3 части:

1. Импортиране на нужни функции и компоненти
2. Кодова част в което се записват нужни за компонента функции и дефиниции
3. JSX част в която се дефинира външния вид и малко логика ако е нужно

JSX (JavaScript XML) е езика чрез който се създават страници в NextJS. Обикновено

файловете са с .jsx, но понеже този проект използва typescript страниците са с .tsx. Както в други системи като .net тези страници са с основа HTML (

Hypertext Markup Language) и за това могат да се използват стандартни HTML елементи като p, div, h1 и други. Важното нещо което JSX позволява е създаване на компоненти и тяхното преизпълване. На Фигура 19 това може да се види. Експортира се компонент наречен Sidebar който съдържа логика и дефиниция за рисуване. В layout на Фигура 14 може да се види как Sidebar е импортиран и използван като стандартен HTML елемент.

По-специфичното за NextJS е използването на ‘use client’ в началото на файла. Понеже NextJS е full stack (предлага frontend и backend) код на едно място трябва специфично да му бъде казано къде този код ще бъде изпълнен. В този случай Sidebar ще бъде изпълнен на клиент (в браузъра на потребителя) и в него не мога да бъде изпълняван код за сървъра или да се вика променлива от околната среда  предназначен само за сървър. В противен случай подразбиране ще се опита да го изпълни на сервъра.

Това също позволява на NextJS да оптимизира страницата преди да я прати на потребителя. Ако няма React код които динамично да създава HTML, няма нужда да бъде зареждан. Просто се генерира кода на сървъра и се изпраща на потребителя.

### **5.3.3 Страница Register**

На страницата register може да се види начина по който се обработват данните от страната на клиента. В стандартна React форма би се използвало useState за да бъде запазено състоянието на определена променлива взета от HTML input. В този проект се използва библиотеката zod. Тя предлага три много удобни неща за един програмист:

* Всяка променлива се съдържа в един обект – форма. Не е нужно създаването на няколко променливи
* Предоставя вградена функционалност за преварване на променливите дали отговарят на определени условия
* Задаването на специфични грешки и тяхното лесно показване

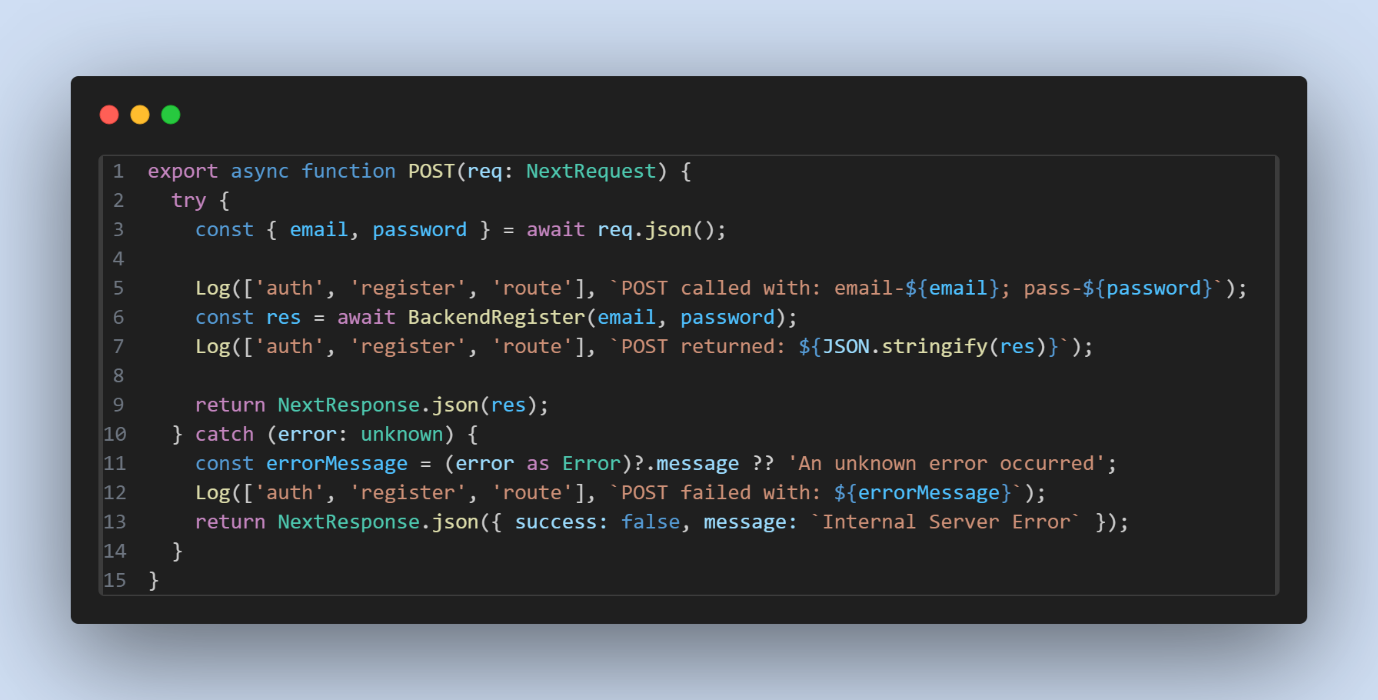
За разлика от други методи като .net няма директен начин за пращане на информация на сървър за това се използва API. Помагащата функция APICaller помага като в първата част се подава локацията от където е викната, това спомага за решаване на бъгове и проблеми, подава се маршрута на който живее API точката която иска да достъпи, HTTP метода за връзка, GET ако се взимат данни, POST ако се пращат данни и, ако е нужно обект с данни. Връща обект с две променлив: success и данни. В зависимост от bool success се определя какъв toast да бъде показан на потребителя и, ако има данни, как да се обработят.

За навигация се използват линкове. В кода е използван компонента Link от Next. Това се прави с цел по-добра работа. Обикновения a таг който се използва в HTML за навигация презарежда напълно страницата. Понеже NextJS използва React той е SPA(Single Page Application – Приложение на една страница). Смененето на страница става чрез промяна на състоянието и прерисуване на екрана с новата информация.



Фигура 20 - Обсъдения код на Register

След като напише и валидира информацията се изпраща на сървър за повече обработка и действие върху тази информация. Започва се в точката искана от APICAller /api/auth/register. Може да се види на Фигура 21. С цел пре използваем код и проста логика тя е много проста. Всяка API точка използва JSON (JavaScript Object Notation) за съдържане на данните при трансфер. В този случай се взимат емайл и парола. Вика се Log за да бъде изписано на логове какво се случва и се вика функцията BackendRegister. Тя извършва цялата работа нужна за регистриране на потребител. Както се вижда цялата тази логика е обвита в try catch. Идеята че тези функции могат да се провалят. Ако нещо такова се случи сървър няма да спре да работи а ще бъде хванат. Отново се записва каква грешка се е случила, а за потребител се връща лесно разбираем съобщение. Както може да се види на Фигура 20. BackendRegister също връща JSON защото една функция може да се провали по два начина. Защото нещо се объркало или защото програмиста е решил че трябва. Тази функция добър пример и може да се види на Фигура 22. Ако се провали поради грешка ще бъде казано на потребителя чрез try catch, но ако има проблем с данните функцията сама ще върне success = false и съобщение което пак ще бъде подадено на потребителя.



Фигура 21 - /api/aith/regsiter/route.ts

Самата функция е сравнително проста. Първо се проверява дали потребителя съществува чрез FindUserByEmail. След това се определя дали потребителя вече има акаунт в системата чрез други снабдители (OAuth). Този проект вярва че не трябва да се вярва на потребителя и че всеки може да направи request към API. За това всички проверки върху емайл и парола за валидност отново се правят. След това или се създава нов потребител с подадените данни или се прикачат към вече създаден от друг снабдител.



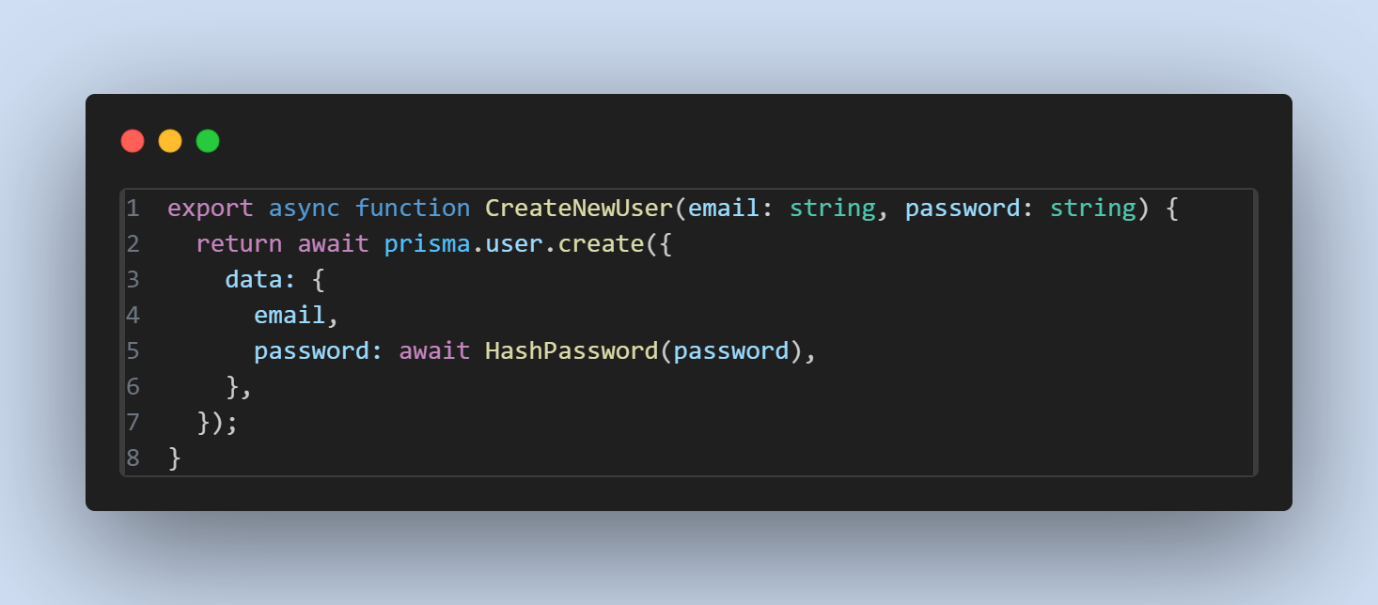
Фигура 22 - BackendRegister в auth-utils.ts

Както може да се види на Фигура 22 има много асинхронни функции които правят нещо специфично. Това отново е направено с цел не повтаряне на код и лесно поддържане. За добър пример може да се разгледа функцията CreateNewUser на Фигура 23. В крайна сметка всички данни в една програма са просто редове в база данни и за връзката се използва Prisma. Prisma е ORM (Object-Relational Mapping - Обектно-релационно картографиране) което предлага три много важни неща:

* Лесна абстрактна работа с база данни и създаване на заявки
* Сигурност на типа
* Предпазване от SQL инжектиране

Prisma абстрахира както SQL (Structured Query Language - Структуриран език за заявки), езика използван за комуникация с базата данни, така и самата връзка. Чрез записване на низа за връзка с базата и кодово, като typescript с типове, описване на структурата, както може да си види на Фигура 23, програмист не е нужно да пише SQL код за заявките си или да се тревожи каква база от данни използва, това е много удобно за проекти нужни да се развиват ида използват по-висши функционалности на бази данни. Когато заявка се направи към базата тя винаги ще върне обект от вече дефиниран тип с ясно променливи. Също така предпазва от SQL инжекция атака. Атака при която потребител използва променливи с цел да напише негова SQL заявка и да извади информация до която не трябва да има достъп.

С всичко това може да се види колко е просто да се създаде потребител. Само една заявка в която се подава емайл и хеширана парола използвайки hash функцията на bcrypt и няколко рунда на слагане на сол за да бъде още по-сигурна и по-важно не обратима с алгоритъма EksBlowfish. Паролите не се съдържат в прост текст в базата. За разлика от криптиране което е обратим процес, хеширането не е. Ако подадена парола съществува трябва да бъде намерена по уникалния и хаш.



Фигура 23 - CreateNewUser в prisma-utils.ts

### **5.2.3 Logout**

Работата на тази страница се разделя на две части – обновяване на състоянието на потребителя и обновяване на бисквитките на сървър. Както вече е споменато клиентът използва react-redux за поддържане на състояние на потребителя и то може да бъде обновено само от строго определени функции. Използвайки куката useDispatch се извиква AuthLogout което маха всякакво състояние от потребителя.

Отново никога не се вярва на потребителя и това е само за удобство на програмиста. Истинското състояние се пази в две бисквитки accessToken и refreshToken. accessToken главно се използва за достъп като съдържа id на потребителя живее 15 мин и е JWT (JSON Web Token), refreshToken живее 7 дни, използва се като accessToken когато трябва да се създаде нов accessToken и се запазва в базата данни. При всяка заявка се проверяват в този ред. Това позволява на потребител да използва сайта без проблем. Ако не влезе в рамките на 7 дни ще бъде изваден от системата за сигурност. Също така се пази и в базата данни с цел да бъде изтрит и бъдещи нападатели да загубят достъп до акаунта ако е нужно. Те биват изтрити от сървър.

Дори ако нещо стане по време на процеса винаги ще бъдат изтрити бисквитките и визуално потребителя ще бъде изваден от системата с toast съобщение.

### **5.2.4 Страница Profile/HistoryDisplay**

Страницата „Профил“ в приложението предоставя на потребителя цялостен преглед на неговата активност и персонализация. Тук се показва информация за текущо вписания потребител, извлечена от auth обекта в глобалния Redux store, заедно с достъп до функционалности като смяна на парола и управление на локални настройки чрез компонента SettingsGrid, който се намира и в страничната навигация (Sidebar). Основният фокус на страницата обаче е модулът за преглед на калкулации – HistoryDisplay.

HistoryDisplay представлява динамична система за показване и филтриране на всички калкулации, които потребителят е запазил, докато е бил вписан. Историята се зарежда от API точката /api/user/calc-history, която агрегира резултати от всички калкулатори в приложението, но показва само тези, които са създадени от текущия потребител. За по-добро потребителско изживяване, калкулациите са групирани по вид и подредени в раздели (tabs), като началният ред е по дата – от най-новите към най-старите.

Системата предлага възможности за филтриране на резултатите чрез търсене по име на растение (на текущо избрания език) или чрез избор на конкретна дата. Това улеснява бързото намиране на стари изчисления, като същевременно дава възможност на потребителя да сравнява стойности през времето, за анализ и самооценка. Всеки запис в историята е представен с визуална карта (Card), която включва входни и изходни стойности от съответната калкулация, придружени от контекстуални диаграми за допълнителна яснота. Това се постига като при промяна



Фигура 24 - HistoryDisplay подава заявка за история и филтрира

### **5.2.5 Страница Sowing**

Страницата за калкулатор на норма сеитба (`/app/calculators/sowing/page.tsx`) представлява една от основните функционалности на приложението, позволявайки на земеделските производители да изчислят оптималната норма за засяване на различни култури. Изчисляването на правилната норма сеитба е критично важно за успешното земеделие, тъй като осигурява икономическа ефективност чрез избягване на излишни разходи за семена или загуби в добива, гарантира оптимален добив чрез правилна плътност на растенията, осъществява естествен контрол на плевелите и води до по-високо качество на продукцията с по-добра устойчивост към стрес. Страницата е изградена като клиентски компонент (`use client`), което я прави компонент изпълняващ се на клиента.

При първоначално зареждане страницата извиква API заявка към `/api/calc/sowing/input` за получаване на данните за растенията от базата данни. След това се запазва в state. Същия алгоритъм може да се наблюдава на Фигура 24.

Потребителят започва с избор на растение от падащо меню, което се попълва динамично от заредените данни. При избор на култура се активират входните полета за параметрите на сеитбата, използвайки компонента BuildSowingRateRow за всеки входен параметър.

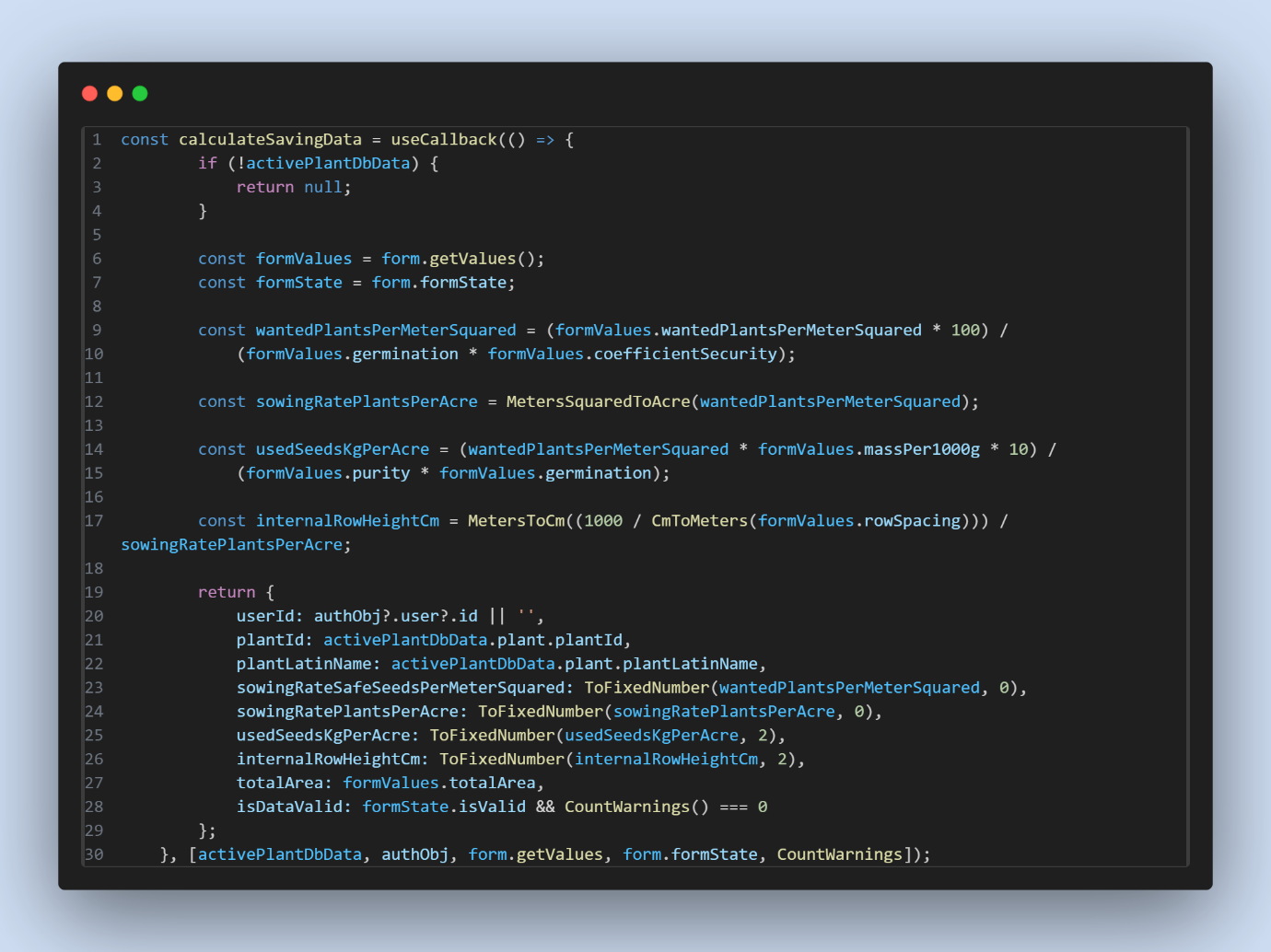
При валидни данни се показват изчислените резултати чрез компонентите SowingOutput – компонента държащ всички изходни данни от калкулатора, SowingTotalArea – понеже калкулатора смята за 1 декар/хектар трябва да се умножът резултатите за да може един фермер да види за неговата желана площ и SowingCharts – където се прави малък и не чак толкова полезен анализ на данните които визуализират различни аспекти от калкулацията. Потребителят може да запази калкулацията чрез бутон, който изпраща данните към API точката `/api/calc/sowing/history` ако е влязъл в системата. Отново никога не се вярва на потребителя и това е защитена точка която не може да бъде достъпена освен ако не е влязъл в системата.

React Hook е специална функция, която дава възможност на функционалните компоненти да използват функции като състояние и ефекти, които преди бяха налични само в класовите компоненти. Всички hooks започват с думата "use" и могат да се извикват само в началото на React функциите - не можем да ги поставяме в цикли, условия или вложени функции. Hooks решават редица проблеми в React разработката. Те позволяват да споделяме логика за състоянието между различни компоненти без да създаваме сложни структури. Освен това помагат да организираме кода според това какво прави, а не според React lifecycle методите. И най-важното - избягваме проблемите с `this` в JavaScript, които често объркват програмистите при работа с класове.

`useSowingRateForm` е централният custom hook за управление на цялата логика на формата за норма сеитба. Този hook демонстрира как сложна бизнес логика може да бъде изнесена извън компонента и да стане преизползваема. Hook-ът използва Zod библиотеката за създаване на схема за валидация на данните. Zod осигурява типова сигурност, автоматична валидация и ясни съобщения за грешки. За управление на формата се използва React Hook Form с Zod resolver, което осигурява ефективна работа без излишни рендирания. Двете най-важни функционалности на тази кука са да запазва препоръчителни данни за растение в state и да слага начални стойности на променливите – Фигура 25. Друга важна функционалност е да пресмята финалния резултат който да бъде показван на потребителя и запазван в системата ако е вписан – Фигура 26.



Фигура 25 - useSowingRateForm при промяна на стойности и тяхното проверяване



Фигура 26 - useSowingRateForm pпресмятане на финални стойности

Друг пример за кука е преизпълване на функционалност. Добър пример за това е useTranslate, но тук се използва и друг добър пример който може да се види на Фигура 27 – useWarning. С цел улесняване на потребителя калкулатора разполага с, вече обсъдени, начални данни и препоръчителни стойности. Докато потребителя е в тях всичко е наред и формата е в състояние OK. Ако се излезе извън тях трябва да се премине в състояние Предупреждение. Това казва на потребителя че данните няма да са точно и не се препоръчителни. И на последно място е състояние Грешка. То трябва да се използва само когато има грешка в данните или има грешка в кода. За съжаление избраните пакети за валидация не подържат състояние Предупреждение. За това чрез тази кука и зададени правила може да се покаже на потребителя че не се препоръчват данните, с променливо съобщение, без да бъде спиран от тяхното записване в историята – Фигура 27.



Фигура 27 - useWarnings

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.

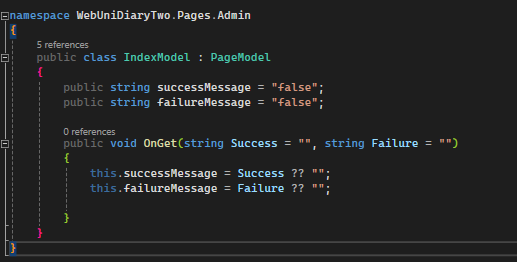
Всички ajax заявки:

* 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.

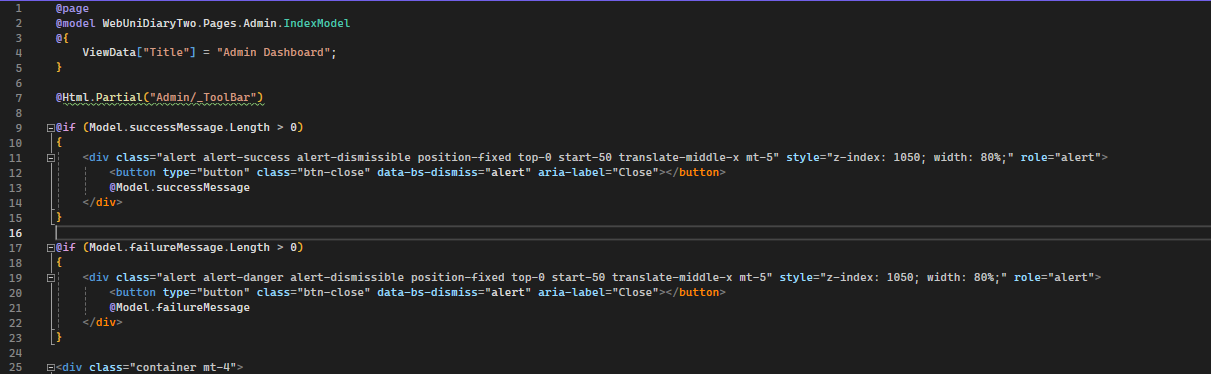
Допълнително главната страница се използва за редирект, при случай че има проблем с приложението. Или е извършено някакво действие успешно, което няма страница за пренасочване.



*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.

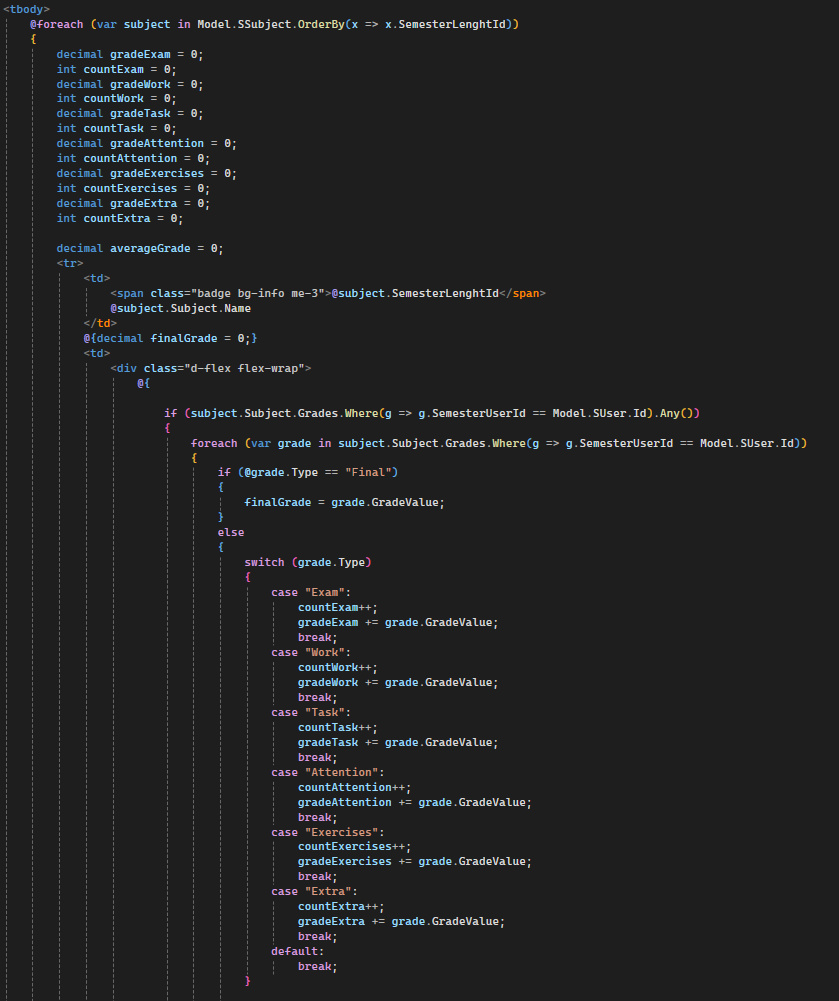
Първо се извличат всички оценки и се инициализират променливите gradeExam, gradeWork, finalGrade и други. При всяко добавяне се брои броят на въведените оценки с помощта на други променливи като countExam, countWork и т.н. След това за всеки тип оценка резултатът се дели на броя оценки, умножава се по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

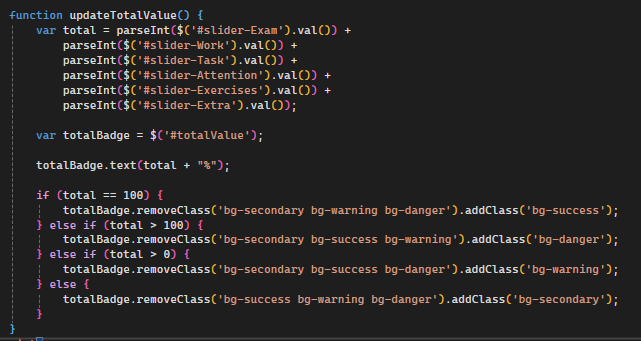
Първо, предметите се сортират по това дали преподавателят е главен или помощен преподавател.

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.

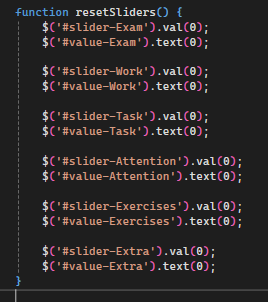
Освен AJAX заявки, на страницата има и няколко помощни JS метода.

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.

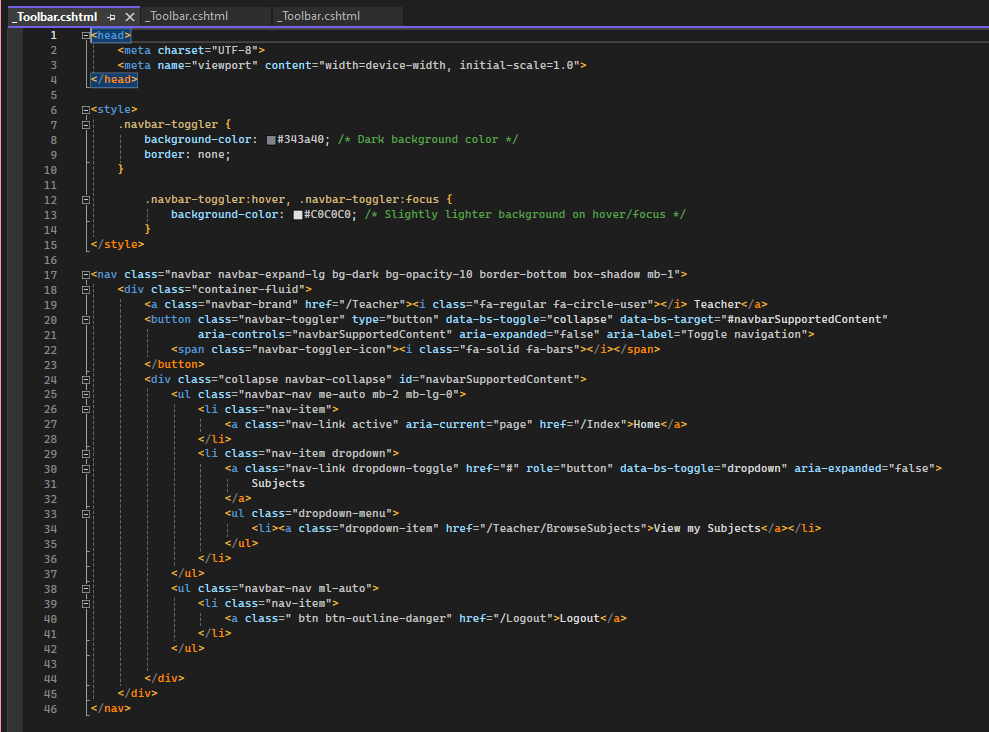
### **5.2.15 Страница Shared**

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*

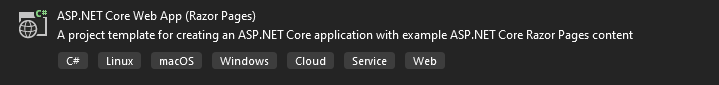
# **6. Ръководство за работа със системата**

## **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*

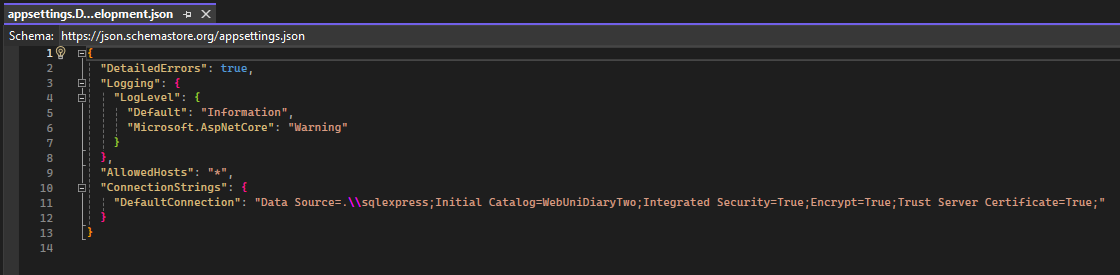
### **6.1.2 Стартиране на проекта**

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.

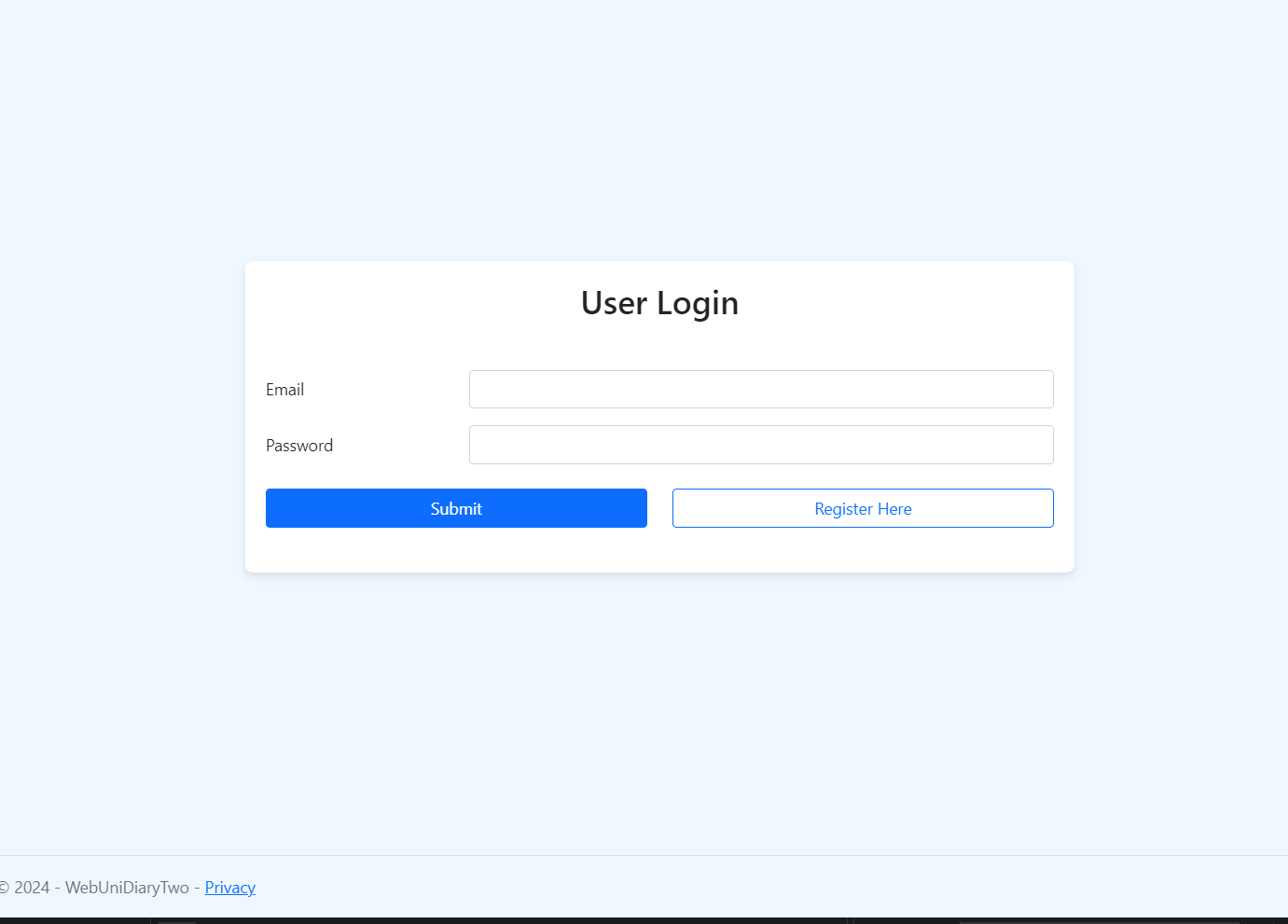
При първото стартиране на SQL Server, трябва да се добави потребител от всеки тип (администратор, преподавател и студент), за да могат потребителските групи да бъдат създадени.



*Фигура 6.2 – Конфигурационни данни*

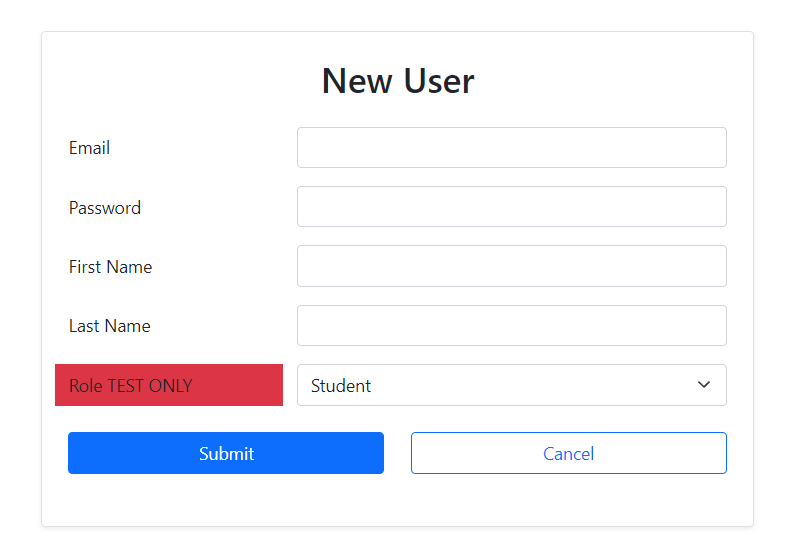
След това може да се стартира поректа. При успех ще се покаже Log In страницата след като се стартира преложението. Фигура 6.3 показва точно това.

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**

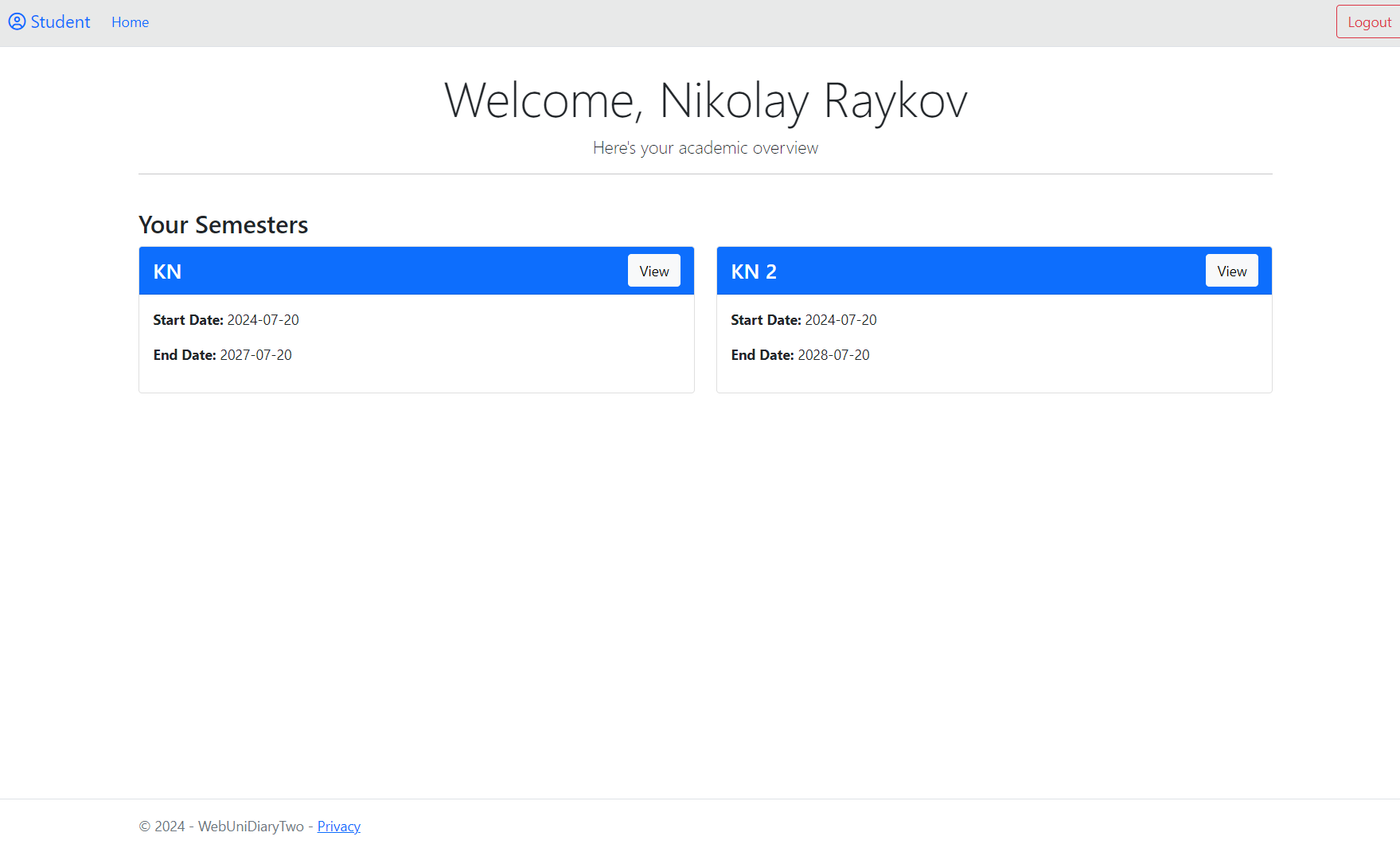


*Фигура 6.4 – Регистрационната форма*

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.

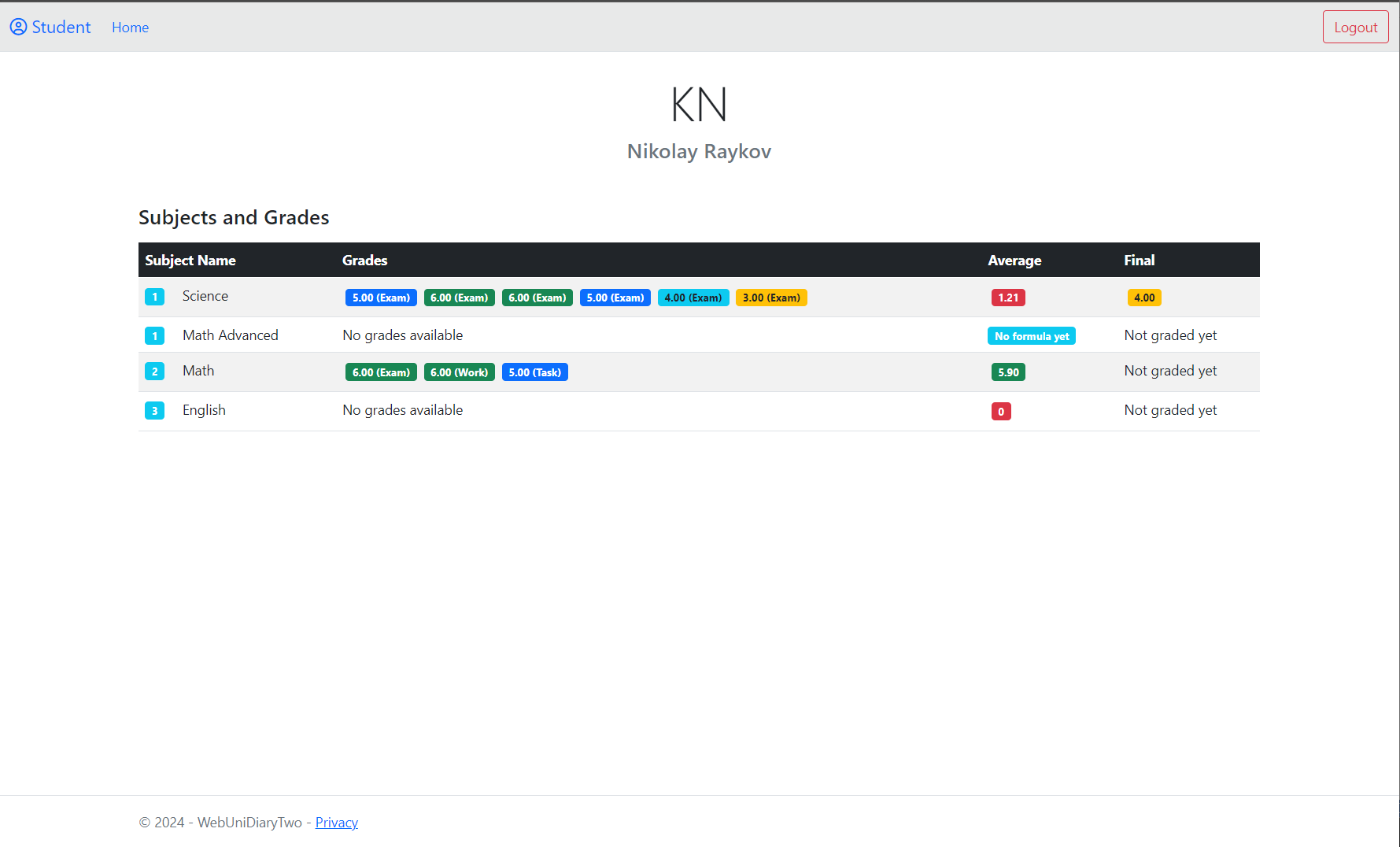


*Фигура 6.5 – Успешно вписан студент, страница Student/Index*

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.

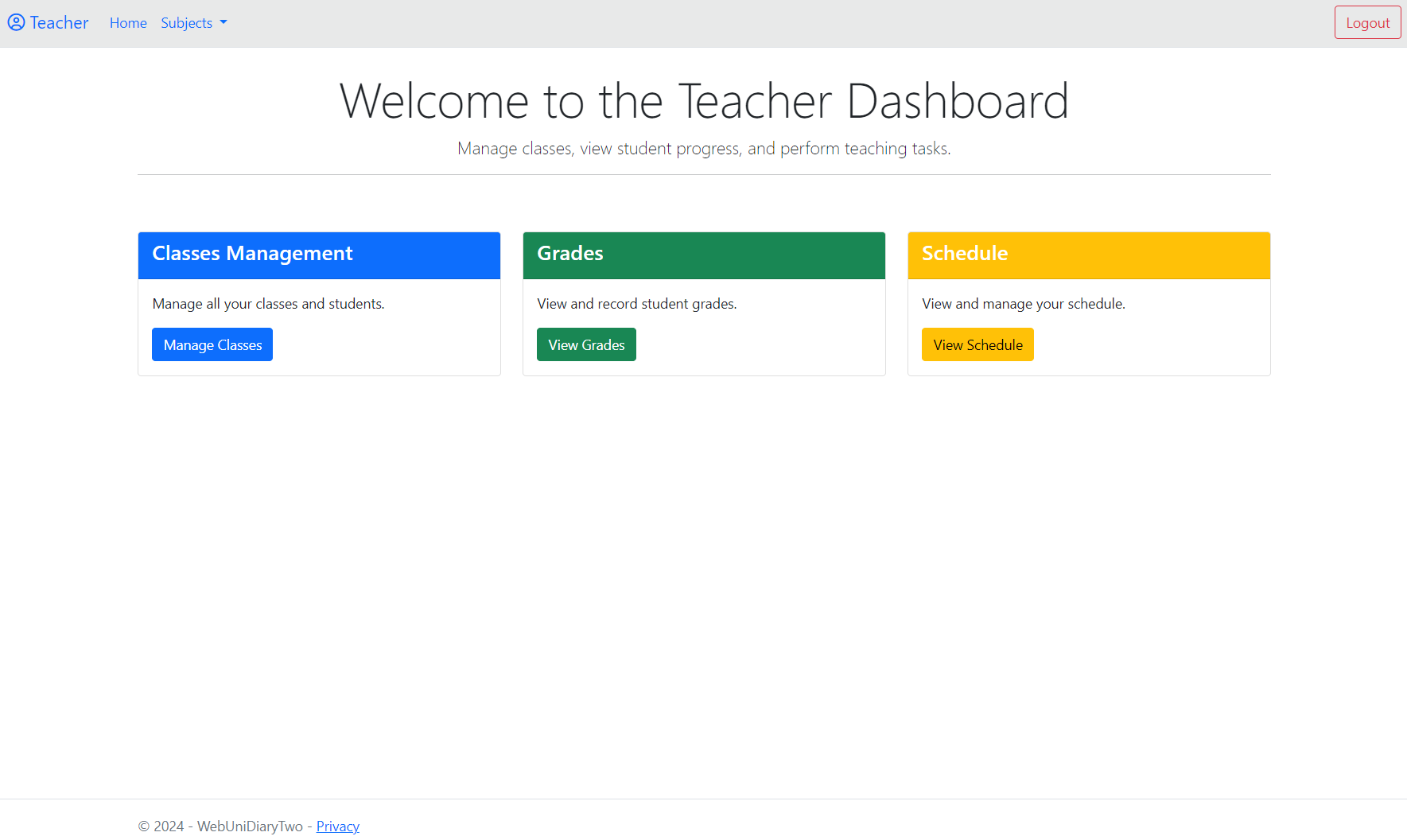


*Фигура 6.6 – SemesterOverview може да се види академичното състояние на студентът*

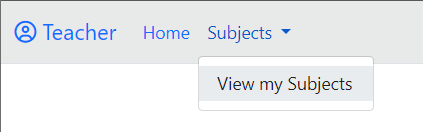
### **6.4 Употреба на приложениетж като преподавател**

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.



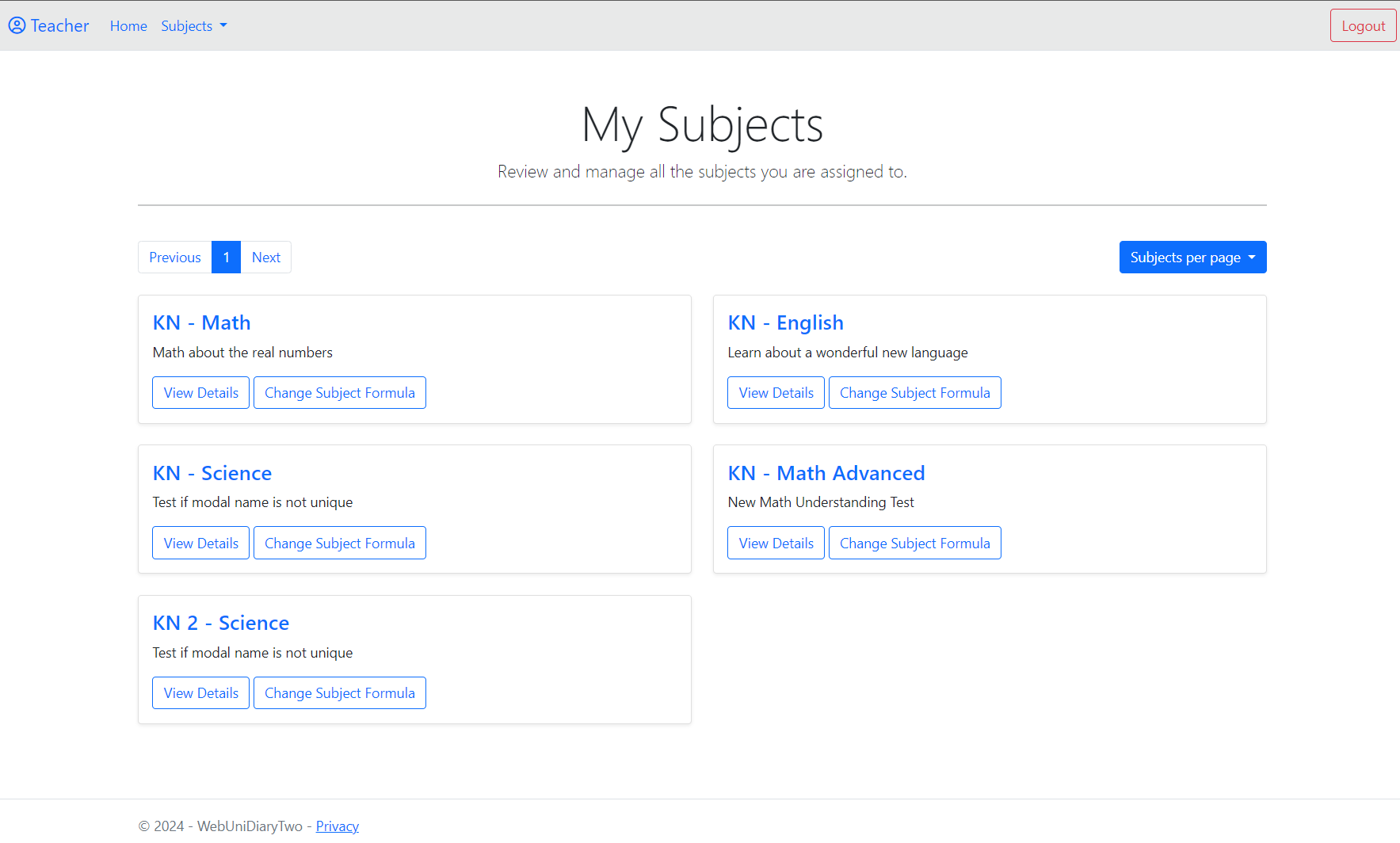
*Фигура 6.7 – Главна страница на Преподавател – Teacher/Index*



*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.

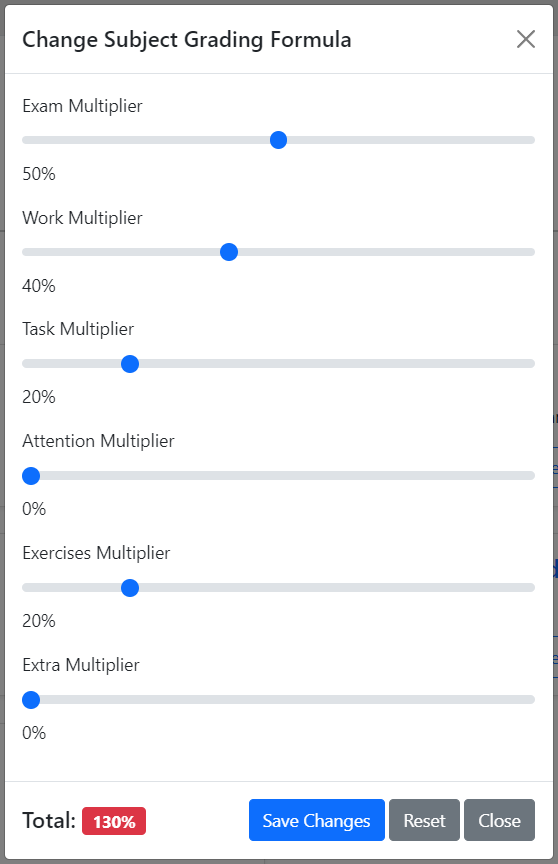
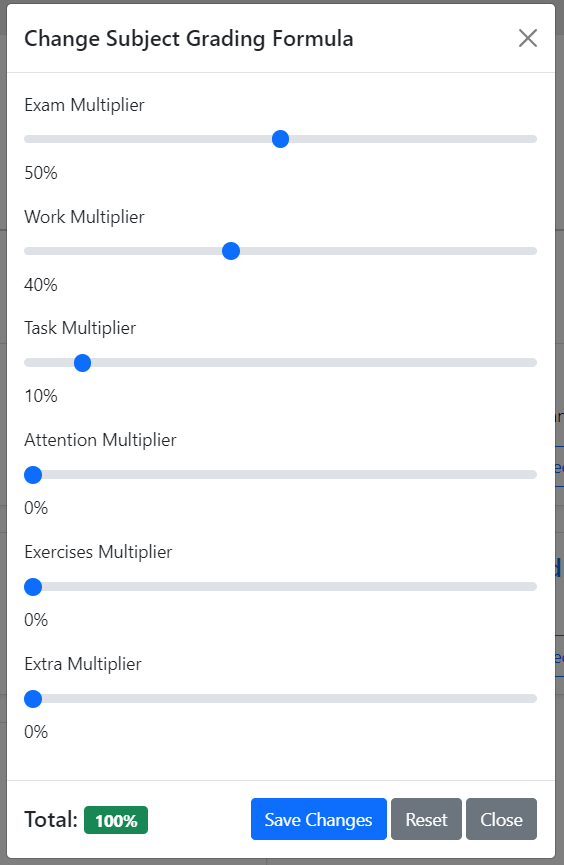
Освен възможността да прегледа всеки предмет, преподавателят може също да променя формулата, с която се изчислява успехът по дисциплината.



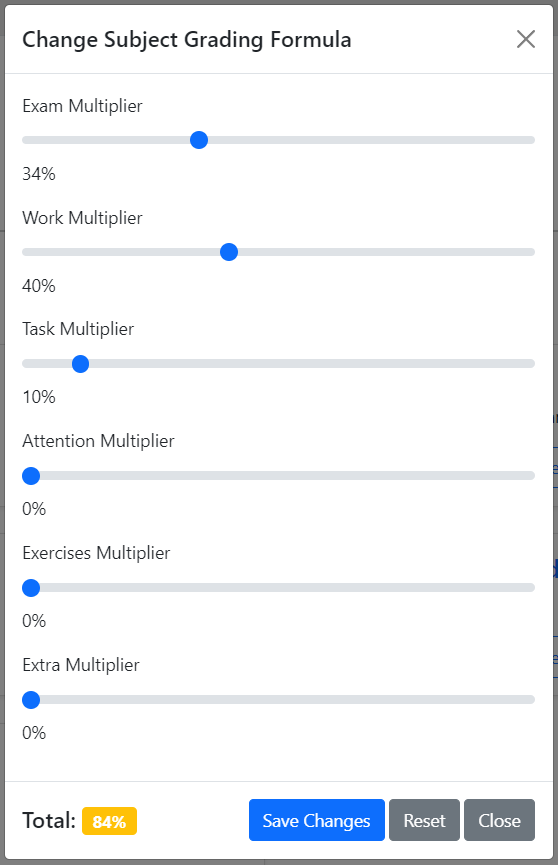
Фигура 6.9 – Страница Teacher/BrowseSubjects

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

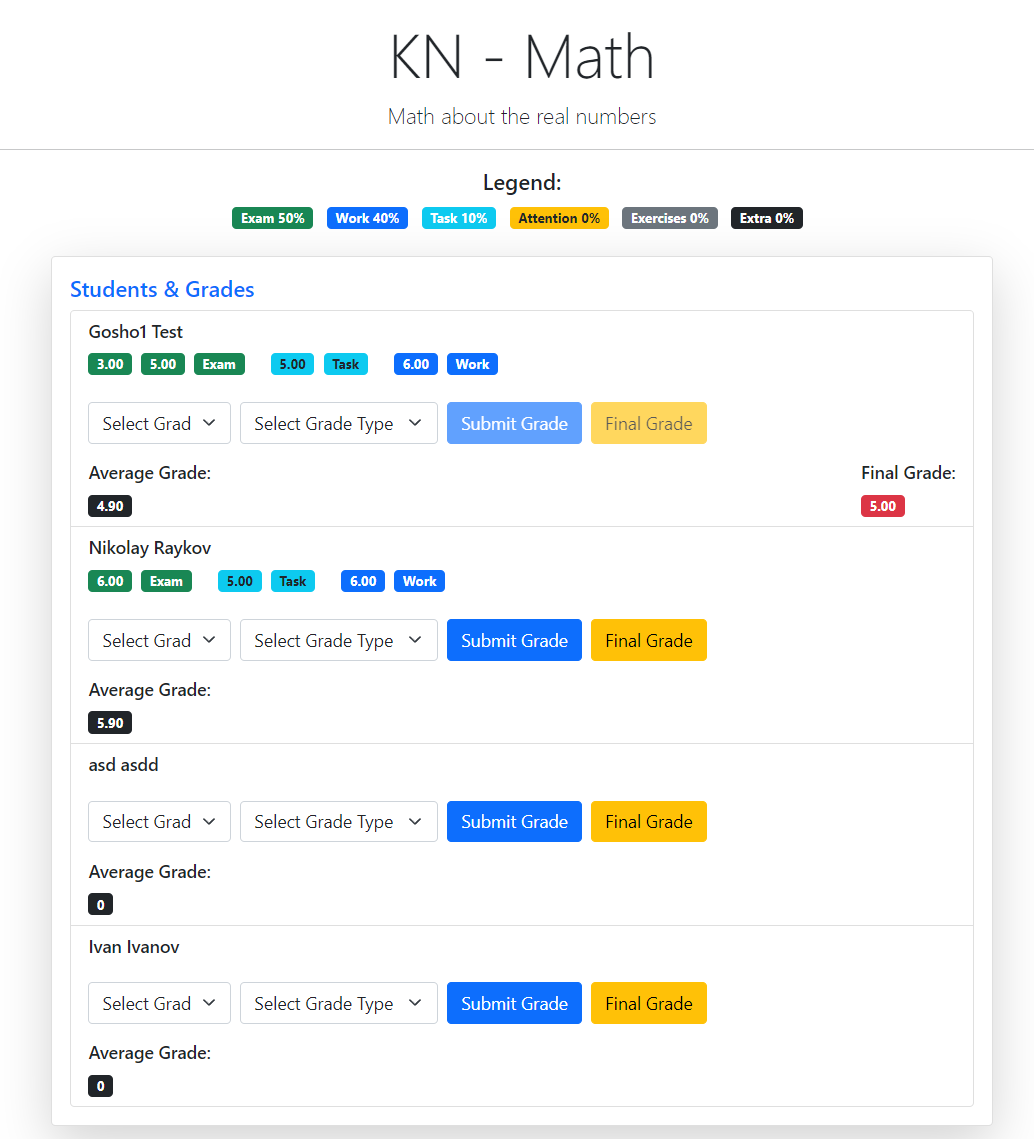
*Фигура 6.10 – Успешна формула Фигура 6.11 – Не успешна формула*

**

*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*

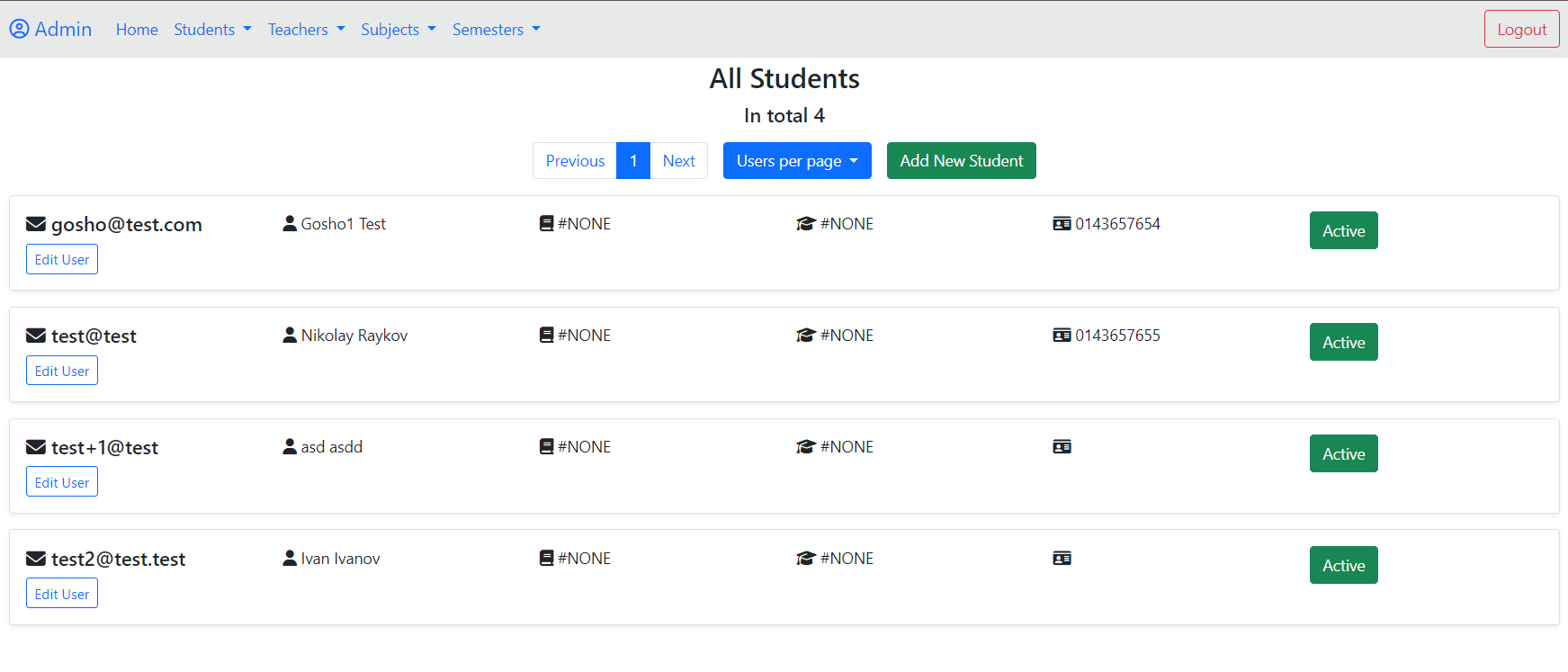
Преподавателят лесно може да избере дадана оценка и да я нанесе стига студентът да няма нанесена крайна оценка.

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

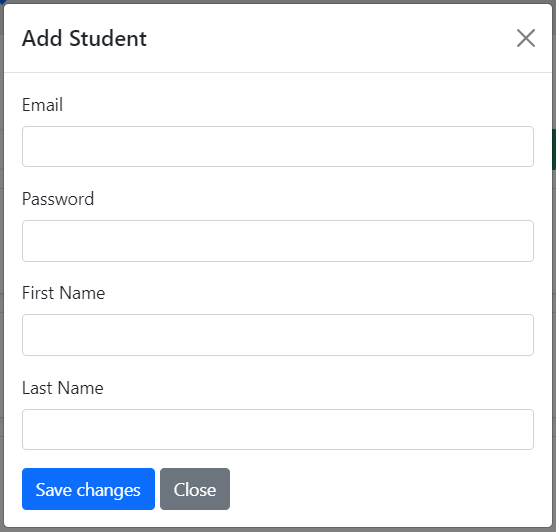
При успешно вписване в системата, администраторът ще бъде приветстван от началната страница Admin/Index. Тук са представени различни линкове за бърз достъп до различни страници.

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.

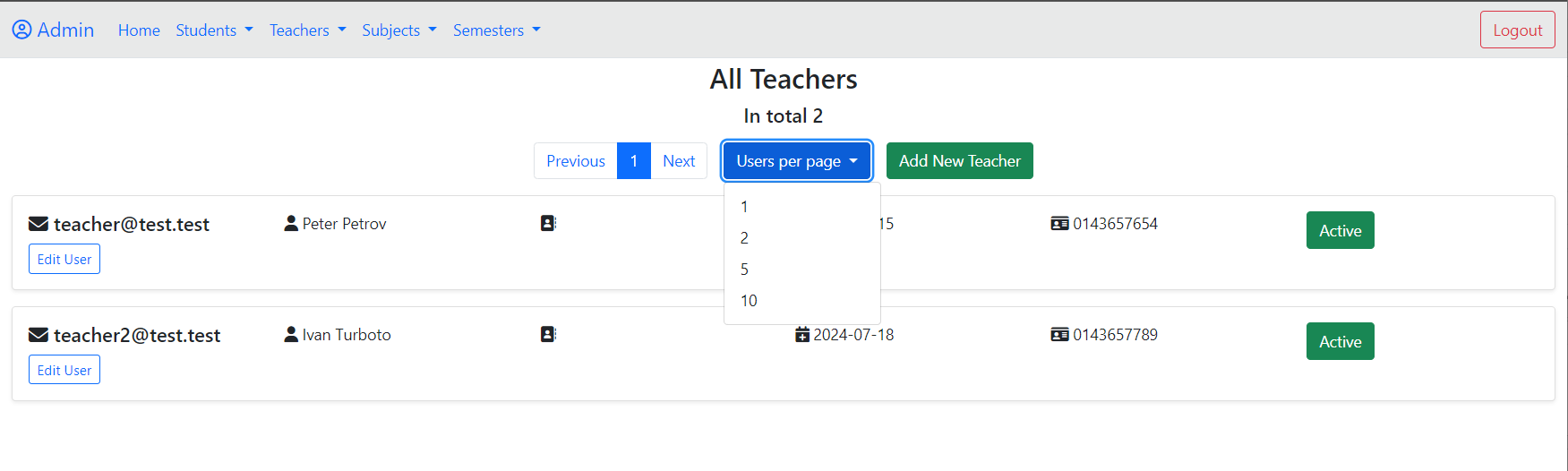


*Фигура 6.14 – Страница Admin/BrowseStudents*



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

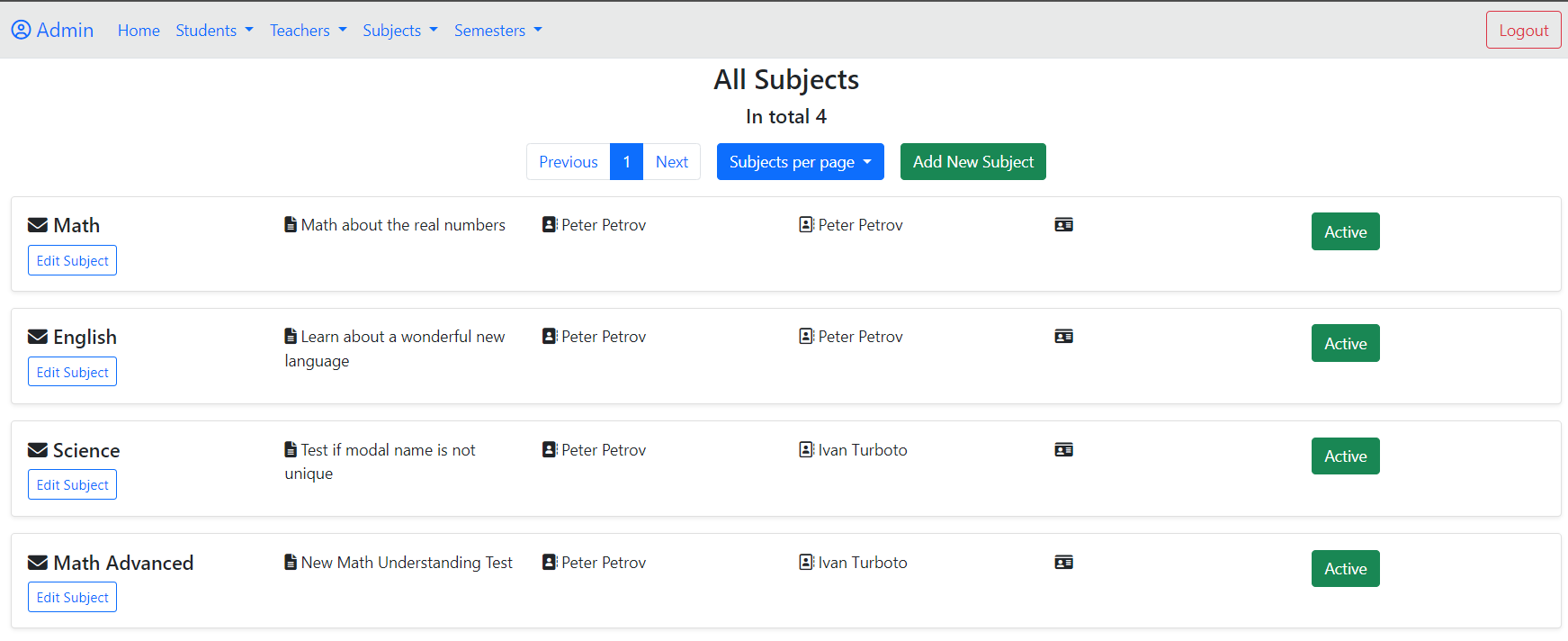
Друга страница, достъпна за администраторите, е тази с всички преподаватели. Тя се намира на **Admin/BrowseTeachers**. Тук се предлага идентична функционалност като при страницата за студентите – добавяне, редактиране и преглеждане на преподаватели.



*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.

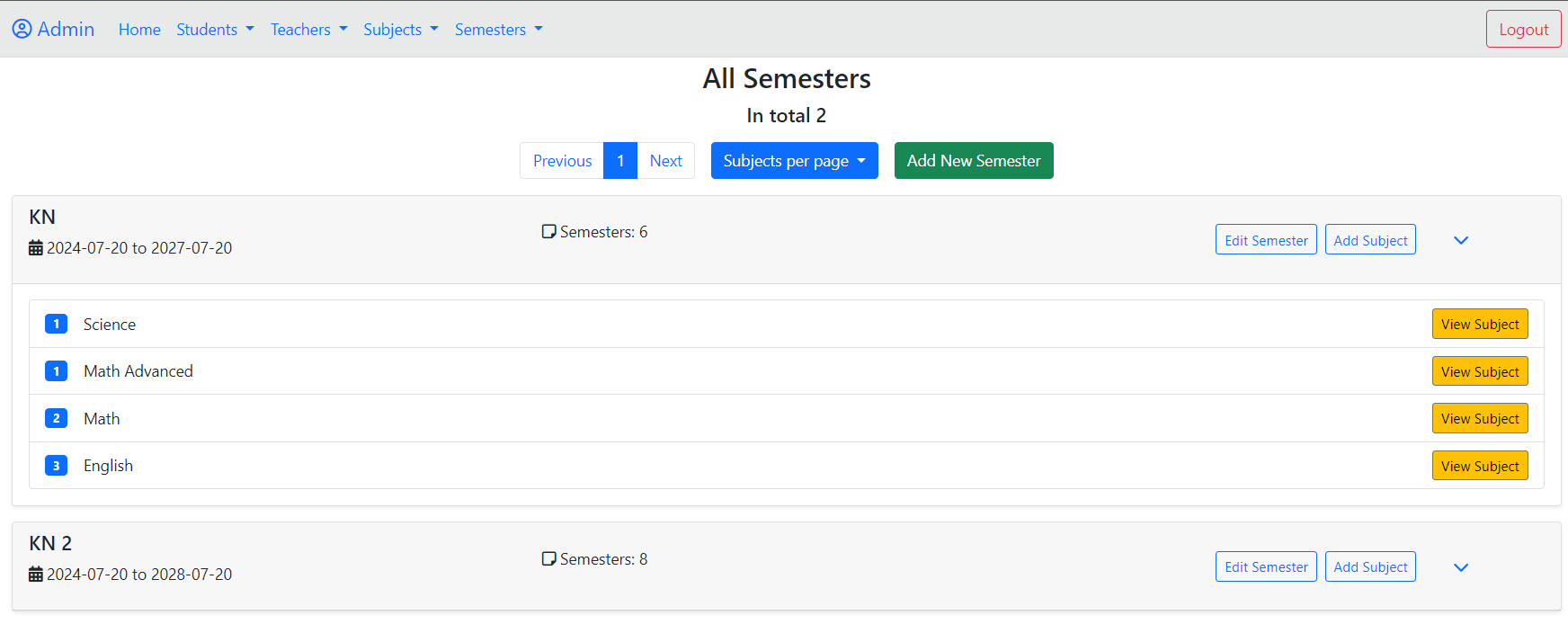


*Фигура 6.17 – Странциа Admin/BrowseSubjects*

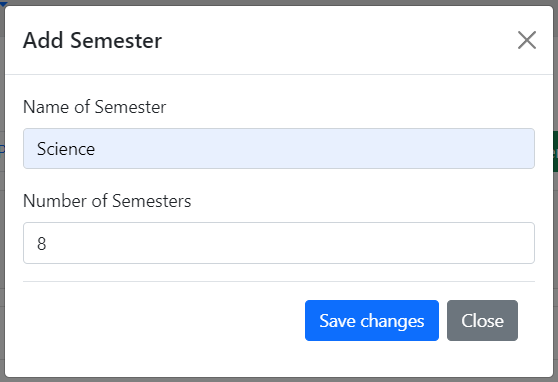
Последната страница, достъпна за администраторите на този етап, е страницата за всички специалности. Тук може да се види всяка специалност и нейният учебен план. Фигура 6.18 показва тази страница.

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.

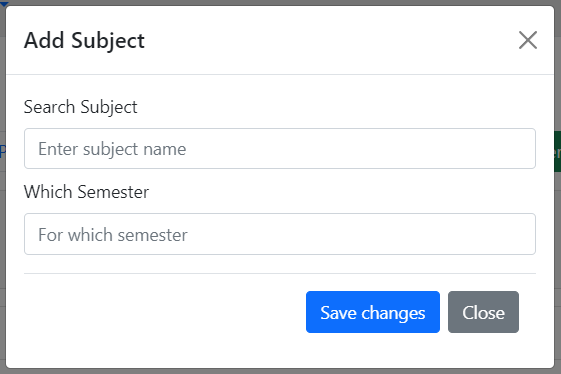
Няколко важни стъпки: специалностите трябва да имат четен брой семестри, в противен случай не могат да бъдат запазени в системата. Действително, не е възможно да има учебна година, състояща се от само един семестър.



*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.

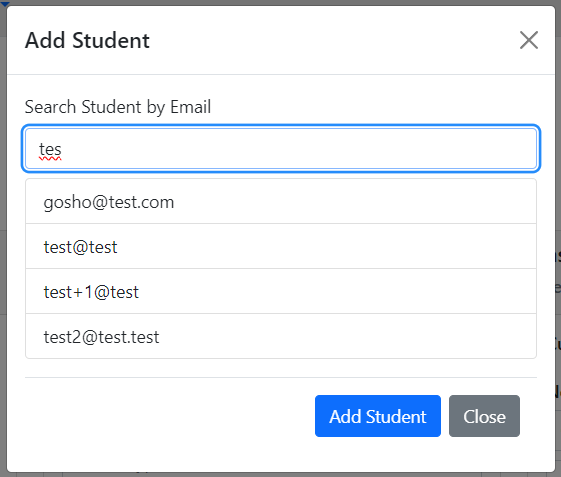
Тук освен да се разглеждат оценките на различните студенти, могат и да се добавят нови, при случай че липсват или е настъпила грешка при преподавателя.

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] | Cope, „tgw-calculator,“ [Онлайн]. Available: https://copeseeds.co.uk/services/tgw-calculator/. [Отваряно на 17 3 2025]. |
| [3] | Реколта.bg, „planove,“ Реколта digital, [Онлайн]. Available: https://digital.rekolta.bg/planove/. [Отваряно на 17 3 2025]. |
| [4] | Agrii, „seed-rate-calculator,“ Agrii, [Онлайн]. Available: https://www.agrii.co.uk/calculators/seed-rate-calculator/. [Отваряно на 17 3 2025]. |
| [5] | 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]. |
| [6] | preparati.info, „raztvor,“ [Онлайн]. Available: https://preparati.info/calculator/raztvor. [Отваряно на 17 3 2025]. |
| [7] | Microsoft, „visualstudio,“ Microsoft, [Онлайн]. Available: https://code.visualstudio.com/. [Отваряно на 17 3 2025]. |
| [8] | L. Torvalds, „git-scm,“ Microsoft, [Онлайн]. Available: https://git-scm.com/. [Отваряно на 17 3 2025]. |
| [9] | Docker, Inc., „Docker,“ Docker, Inc., [Онлайн]. Available: https://www.docker.com/. [Отваряно на 17 3 2025]. |
| [10] | Meta Platforms, Inc., „React,“ [Онлайн]. Available: https://react.dev/. [Отваряно на 11 6 2025]. |
| [11] | Vercel, „nextjs.org,“ Vercel, [Онлайн]. Available: https://nextjs.org/. [Отваряно на 17 3 2025]. |
| [12] | Prisma Data, Inc., „Prisma,“ Prisma Data, Inc., [Онлайн]. Available: https://www.prisma.io/. [Отваряно на 17 3 2025]. |
| [13] | The PostgreSQL Global Development Group, „postgresql,“ The PostgreSQL Global Development Group, [Онлайн]. Available: https://www.postgresql.org/. [Отваряно на 17 3 2025]. |
| [14] | Microsoft, „typescriptlang,“ Microsoft, [Онлайн]. Available: https://www.typescriptlang.org/. [Отваряно на 17 3 2025]. |
| [15] | D. Abramov и A. Clark, „react-redux,“ [Онлайн]. Available: https://react-redux.js.org/. [Отваряно на 18 3 2025]. |
| [16] | C. McDonnell, „zod,“ [Онлайн]. Available: https://zod.dev/. [Отваряно на 18 3 2025]. |
| [17] | OpenJS Foundation, „jestjs,“ [Онлайн]. Available: https://jestjs.io/. [Отваряно на 18 3 2025]. |
| [18] | shadcn, „shadcn,“ [Онлайн]. Available: https://ui.shadcn.com/. [Отваряно на 18 3 2025]. |
| [19] | Tailwind Labs Inc., „tailwindcss,“ [Онлайн]. Available: https://tailwindcss.com/. [Отваряно на 18 3 2025]. |
| [20] | E. Fennis, K. Rigó и J. Guddas, „icons,“ [Онлайн]. Available: https://lucide.dev/. [Отваряно на 18 3 2025]. |
| [21] | 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]. |
| [22] | Microsoft, „github,“ [Онлайн]. Available: https://github.com/. [Отваряно на 18 3 2025]. |
| [23] | Microsoft, „actions,“ [Онлайн]. Available: https://github.com/features/actions. [Отваряно на 18 3 2025]. |
| [24] | netcup GmbH, „vsp,“ [Онлайн]. Available: https://www.netcup.com/en/server/vps. [Отваряно на 18 3 2025]. |
| [25] | Cloudflare, Inc., „cloudflare,“ [Онлайн]. Available: https://www.cloudflare.com/. [Отваряно на 18 3 2025]. |

# **9. Таблица с фигури**

[Фигура 1 Cope 6](#_Toc204121870)

[Фигура 2. Реколта digital 7](#_Toc204121871)

[Фигура 3 Agrii 8](#_Toc204121872)

[Фигура 4 agrocalculator.bg 9](#_Toc204121873)

[Фигура 5 preparati.info 10](#_Toc204121874)

[Фигура 6 Диаграма на случаите на употреба 18](#_Toc204121875)

[Фигура 7 Диаграма на дейностите – създаване на нова специалност 19](#_Toc204121876)

[Фигура 8. Релационен модел на базата данни 20](#_Toc204121877)

[Фигура 9 - Файлова структура - A 26](#_Toc204121878)

[Фигура 10 Файлова структура - B 26](#_Toc204121879)

[Фигура 11 - Файлова структура C 27](#_Toc204121880)

[Фигура 12 - Файлова структура D 27](#_Toc204121881)

[Фигура 13 - Папката /hooks 28](#_Toc204121882)

[Фигура 14 - Файл layout.tsx 29](#_Toc204121883)

[Фигура 15 - instrumentation.tsx 30](#_Toc204121884)

[Фигура 16 - middleware.ts 32](#_Toc204121885)

[Фигура 17 - store.ts 34](#_Toc204121886)

[Фигура 18 - Как точно изглежда една типична .tsx страница 35](#_Toc204121887)

[Фигура 19 - Обсъдения код на Sidebar 36](#_Toc204121888)

[Фигура 20 - Обсъдения код на Register 38](#_Toc204121889)

[Фигура 21 - /api/aith/regsiter/route.ts 39](#_Toc204121890)

[Фигура 22 - BackendRegister в auth-utils.ts 40](#_Toc204121891)

[Фигура 23 - CreateNewUser в prisma-utils.ts 41](#_Toc204121892)

# **10. Таблица с таблици**

[Таблица 1 Сравнение на функционалности 11](#_Toc204121893)