

**Профилирана природо-математическа гимназия "Никола Обрешков"**

гр. Казанлък, ул. "Орешака" 2, тел.:0431/62334, e-mail: info-2400262@edu.mon.bg

**ДИПЛОМЕН**

**ПРОЕКТ**

**Тема: : РАЗРАБОТКА НА УЕБ ПРИЛОЖЕНИЕ ЗА ГОТВАРСКИ РЕЦЕПТИ „КУЛИНАРНИ ИЗКУШЕНИЯ“**

*Ученик:* ***Креса Юлиан Цветкова***

***Професия:*** *код 481030 „Приложен програмист“*

***Специалност:*** *код 4810301 „Приложно програмиране“*

**Консултант:** Здравка Димитрова

Казанлък, 2024 г.

Увод

1. Увод

* След смяната на учебната програма следваше и промяна в изпитите за завършване на средно образование. Въведен беше ДЗИ по предмет от профилираната подготовка, който се води матура второ равнище. След проведените вече четири сесии с матура по информатика се установи, че много ученици срещат затруднения с намирането на материали за подготовка и за проверка на текущите им знания.
* Така реших да разработя уеб приложение, което да помогне за успешното справяне с матурата по информатика и да се превърне в ефективен инструмент за учене и трениране, който да помогне на учениците да се подготвят по най-адекватния начин за изпита. Чрез моя проект „БлицКод-ДЗИ“ ученици и учители от цяла България могат да контактуват с цел по-добра подготовка за ДЗИ по информатика. Този модел на обучение, който включва споделяне на материали от учителите и самостоятелната подготовка на учениците с множество различни ресурси, е изключително полезен за разнообразяване на образователния процес и за осигуряване на обширен поглед върху необходимите знания и компетентности.

1. Цели
   * Приложението трябва да е с добре организирана трислойна архитектура- да съдържа слой за данни, за бизнес логика и презентационен слой
   * Приложението трябва да реализира CRUD операции за обработка на учебните материали, за работа с форума, за новините на МОН
   * Трябва да се създаде възможност за разглеждане на уроците и упражненията в учебната програма, разглеждане на форума, на потребителските акаунти, на новините на МОН и на потребителските ревюта
   * Най-съществената функционалност ще бъде създаването и разглеждането на уроци и упражнения в учебната програма, качването на всякакъв вид материали(файлове), създаването и разглеждането на постове и въпроси във форума, оставянето на ревю за приложението, създаване, редактиране и разглеждане на МОН новини
   * Трябва да се реализира филтриране по :

* В учебната програма : по модули и секции се филтрират уроците и упражненията.
* Във форума: в зависимост дали поста е въпрос или не
* В администраторския панел: потребителите се филтрират по роля на потребителя
  + Потребителят трябва да може да се регистрира и да получава достъп чрез попълване на потребителско име, парола и съответните данни които се изискват за съответната роля
  + Ясно да се разграничат разрешените дейностите на не регистриран потребител, учител, ученик. Потребител на МОН и администратор

1. Задачи

* За да се изпълнят поставените цели за реализирането на WEB приложението, организиращо дейността на БлицКод-ДЗИ, трябва да бъдат изпълнени следните задачи:
  + Приложението да се раздели на отделни проекти: Презентационен слой, бизнес логика, база от данни и тестове
  + Да се даде възможност на администратора да трие потребители, да дава на учителите права и да създава потребители на МОН, за да се осигури подържане на актуална информация и проследяване на дейността на потребителите;
  + Уеб приложението да осигурява интуитивен за потребителите интерфейс с добре организирани менюта и форми за въвеждане на потребителска информация

1. **Първа глава – Проучване**

С нарастването на цифровата технология се увеличава и значението на уеб приложенията във всички области на живота, включително образованието. Бързият и лесен достъп до образователни материали от всяко място и по всяко време се превръща в ключов фактор за съвременното обучение. В днешно време, реализирането на дейности, свързани с образование и обучение, без наличие на уеб приложения, може да доведе до ограничаване на достъпа до знания и съответно до намаляване на обучаващата ефективност. Поради тази причина, все повече институции и образователни организации се объркат към уеб приложенията, за да осигурят на своите ученици и служители удобство и ефективност в образователния процес.

Съществуват множество уеб приложения, които предлагат възможности за обучение и управление на образователни процеси. Google Classroom е един от най-известните примери, който предоставя платформа за създаване на виртуални класни стаи, качване на материали и даване на задачи за учениците. Moodle също е мощен инструмент за управление на образованието, позволяващ на учителите да създават онлайн курсове с различни видове интерактивни материали. Khan Academy е популярна сред потребителите заради своите безплатни образователни ресурси, които са достъпни за всеки. Edmodo предлага социални функции и виртуални класни стаи за взаимодействие между ученици и учители.

Въпреки че съществуващите уеб приложения предлагат различни възможности за обучение и управление на образователни процеси, те също имат своите предимства и недостатъци. Например, Google Classroom е лесен за използване и интегриран с други Google услуги, но може да е ограничен в някои функционалности. Moodle е многофункционален, но изисква допълнителни умения за настройка и управление. Khan Academy предлага безплатни ресурси, но понякога липсва персонализиран подход за учениците. Edmodo осигурява социални възможности, но може да е по-сложен за навигация в сравнение с другите платформи.

БлицКод-ДЗИ се отличава от съществуващите уеб приложения по няколко ключови аспекта. Преди всичко, то предлага различни роли за потребителите, което позволява по-голяма гъвкавост в управлението на образователните процеси. Освен това, приложението се фокусира специално върху учебни материали и задачи, което го прави подходящо за учебни заведения и учители. Структурираната информация и възможността за взаимодействие между ученици и учители също са ключови характеристики на приложението.

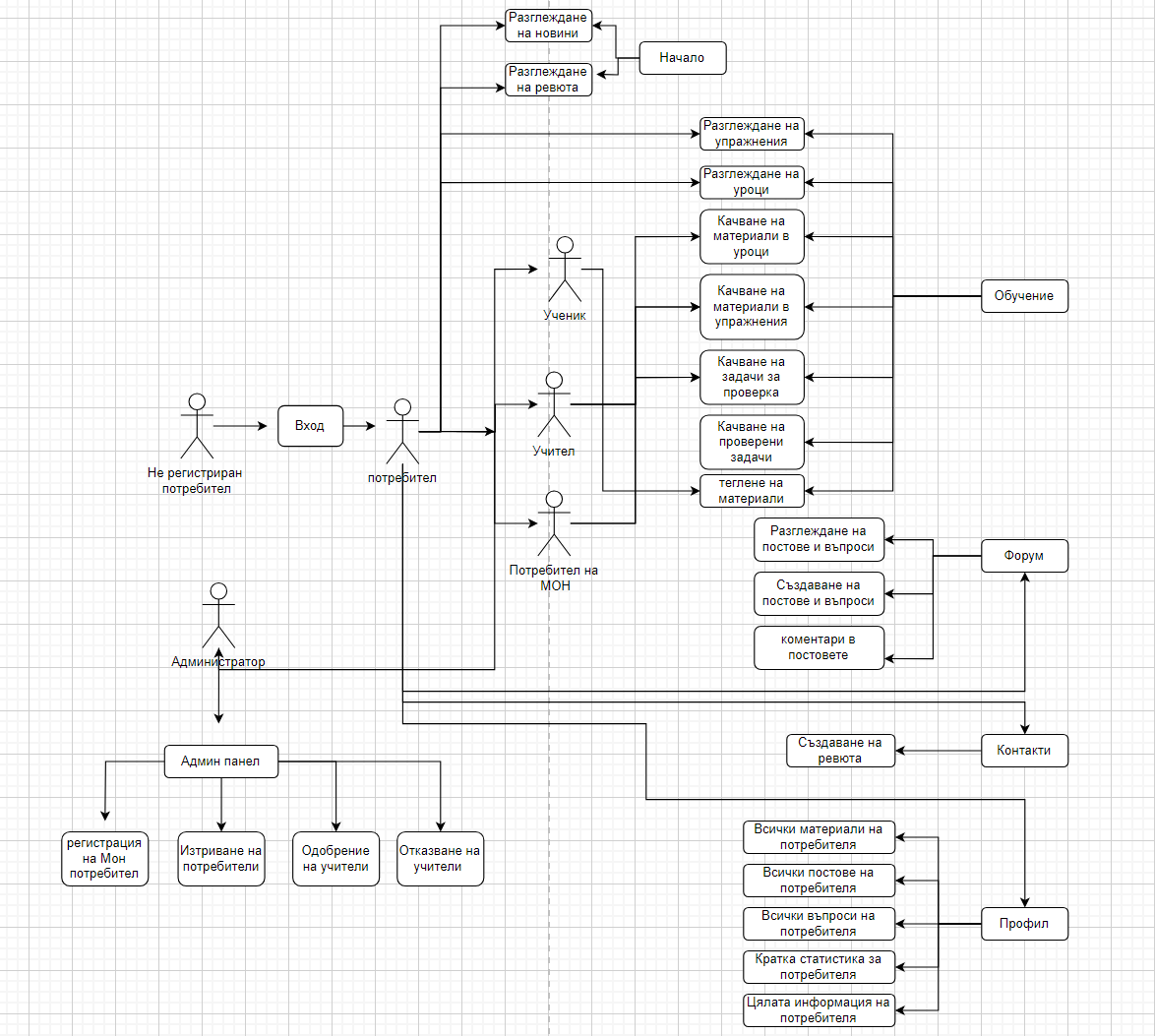
Според последните статистически данни, пазарът на уеб приложения за образователни цели продължава да расте. Google Classroom и Moodle продължават да бъдат водещи в тази област, като за последните години са регистрирани стабилни увеличения на потребителите. Като нов играч на пазара, blickod.pmgkk.com има възможността да се конкурира чрез иновационни функционалности и персонализиран подход към потребителите.

Образователните уеб приложения играят ключова роля в съвременното образование, като предоставят удобство и ефективност в обучението. Вашето уеб приложение, blickod.pmgkk.com, представлява иновативен подход към управлението на образователните процеси и има потенциал да привлече голям брой потребители чрез своите уникални функционалности и структурирана информация. С постоянство и иновации в развитието си, вашето приложение може да допринесе за подобряването на образователните практики и да се утвърди на пазара като важен играч в областта на образователните технологии.

1. **Втора глава – Проектиране**
   1. Функционалности на приложението

Приложението поддържа следните функционалности:

* Създадени са роли:
* Администратор
* Ученик
* Учител
* Потребител на Мон
* Регистрира потребители на МОН с роля администратор
* Име
* Фамилия
* Имейл
* Парола
* Телефонен номер
* Град
* Профилна снимка
* Трие потребители /с роля администратор/
* Одобрява учители /с роля администратор/
* Отказване на учителска роля /с роля администратор/
* Извличане на информация за всички потребители
* Изпращане материал на ученик /с роля учител/
* Изтриване на изпратен материал до ученик /с роля учител/
* Създаване на коментар
* Съдържание
* Снимки
* Изтриване на коментар
* Извличане на учебната програма
* Извличане на упражнение с конкретно id
* Извличане на всички новини
* Извличане на всички ревюта
* Създаване на ревю
* Описание
* Звезди
* Потребителско име
* Извличане на урок с конкретно id
* Създаване на материал за урок /с роля потребител на МОН/
* Създаване на материал за упражнение /с роля потребител на МОН/
* Създаване на материал за урок /с роля учител/
* Създаване на материал за упражнение /с роля учител/
* Качване на задачи за проверка /с роля ученик/
* Качване на проверени задачи/с роля учител/
* Триене на материал от урок /с роля учител/
* Триене на материал от упражнение /с роля учител/
* Създаване на новина
* Заглавие
* Описание
* Полезен линк
* Дата
* Промяна на новина
* Изтриване на новина
* Извличане само на постове
* Извличане само на въпроси
* Извличане на всички постове и въпроси
* Създаване на пост
* Заглавие
* Късо описание
* Дълго описание
* Полезен линк
* Снимки
* Въпрос ли е
* Промяна на пост
* Извличане цялата информация на пост
* Изтриване на пост
* Извличане цялата информация за потребител в профилна страница
* Извличане всички секции на учебната програма
  1. Use-Case диаграма



1. Трета глава – Програмна реализация

(Фиг 1)

**Описание на случаите на Use Case диаграмата (Фиг. 1):**

* Случай на употреба: разглеждане на новини

Цел: Задейства се, когато се отвори началната страница

Категория: Четене

Предусловия: Вход в системата

Успешни постусловия: Успешно четене на новини

Актьори: Всички аутентикирани потребители

* Случай на употреба: разглеждане на ревюта

Цел: Задейства се, когато се отвори началната страница

Категория: Четене

Предусловия: Вход в системата

Успешни постусловия: Успешно четене на ревюта

Актьори: Всички аутентикирани потребители

* Случай на употреба: разглеждане на упражнения

Цел: Задейства се, когато се отвори конкретно упражнение

Категория: Четене

Предусловия: Вход в системата

Успешни постусловия: Успешно четене на упражнение и достъп до материалите в него

Неуспешни постусловия: Съобщение, че упражнението няма информация в него

Актьори: Всички аутентикирани потребители

* Случай на употреба: разглеждане на урок

Цел: Задейства се, когато се отвори конкретен урок

Категория: Четене

Предусловия: Вход в системата

Успешни постусловия: Успешно четене на урок и достъп до материалите в него

Неуспешни постусловия: Съобщение, че урока няма информация в него

Актьори: Всички аутентикирани потребители

* Случай на употреба: Качване на материали в урок

Цел: Потребителя качва избрания материал в конкретен урок

Категория: Създаване

Предусловия: Вход в системата като Потребител на МОН или Учител

Успешни постусловия: Успешно качване на материал в урока

Актьори: Потребител на Мон или Учител

* Случай на употреба: Качване на материали в упражнение

Цел: Потребителя качва избрания материал в конкретно упражнение

Категория: Създаване

Предусловия: Вход в системата като Потребител на МОН, учител

Успешни постусловия: Успешно качване на материал в упражнението

Актьори: Потребител на Мон, Учител

* Случай на употреба: Качване на задачи за проверка

Цел: Потребителя качва избрания материал в конкретно упражнение, за да бъде достъпно за проверка

Категория: Създаване

Предусловия: Вход в системата като ученик

Успешни постусловия: Успешно качване на материал в упражнението

Актьори: Ученик

* Случай на употреба: Качване на проверени задачи

Цел: Потребителя качва проверените задачи в конкретно упражнение, като ги връща на конкретния потребител

Категория: Създаване

Предусловия: Вход в системата като учител

Успешни постусловия: Успешно връщане на проверените задачи на конкретния ученик

Актьори: Учител

* Случай на употреба: Теглене на задачи

Цел: Потребителя тегли вече качен материал

Категория: Основна

Предусловия: Вход в системата

Успешни постусловия: Успешно изтегляне на конкретен материал

Актьори: аутентикиран потребител

* Случай на употреба: разглеждане на постове и въпроси

Цел: Задейства се, когато се отвори форума

Категория: Четене

Предусловия: Вход в системата

Успешни постусловия: Успешно четене на въпроси и постове

Актьори: Всички потребители

* Случай на употреба: Създаване на постове и въпроси

Цел: Задейства се, когато се кликне бутона създай

Категория: създаване

Предусловия: Вход в системата

Успешни постусловия: Успешно създаване на пост или въпрос

Актьори: Всички аутентикирани потребители

* Случай на употреба: Създаване на коментар в пост или въпрос

Цел: Задейства се, когато е отворен конкретен пост или въпрос и се кликне бутона за създаване на коментар

Категория: създаване

Предусловия: Вход в системата

Успешни постусловия: Успешно създаване на коментар под пост или въпрос

Актьори: Всички аутентикирани потребители

* Случай на употреба: Създаване на ревю

Цел: Задейства се, когато е отворена страницата с коментари и се кликне бутона създай

Категория: създаване

Предусловия: Вход в системата

Успешни постусловия: Успешно създаване на ревю

Актьори: Всички аутентикирани потребители

* Случай на употреба: Всички материали на потребителя

Цел: Задейства се, когато е отворена профилната страница.

Категория: основна

Предусловия: отваряне на профилната страница

Успешни постусловия: Достъп до всички материали на потребителя

Актьори: Всички аутентикирани потребители

* Случай на употреба: Всички постове на потребителя

Цел: Задейства се, когато е отворена профилната страница.

Категория: основна

Предусловия: отваряне на профилната страница

Успешни постусловия: Достъп до всички постове, които е качил потребителя

Актьори: Всички аутентикирани потребители

* Случай на употреба: Всички въпроси на потребителя

Цел: Задейства се, когато е отворена профилната страница.

Категория: основна

Предусловия: отваряне на профилната страница

Успешни постусловия: Достъп до всички въпроси на потребителя

Актьори: Всички аутентикирани потребители

* Случай на употреба: Кратка статистика за потребителя

Цел: Задейства се, когато е отворена профилната страница.

Категория: основна

Предусловия: отваряне на профилната страница

Успешни постусловия: Визуализация на броя коментари, постове, въпроси на потребителя

Актьори: Всички аутентикирани потребители

* Случай на употреба: Цялата информация на потребителя

Цел: Задейства се, когато е отворена профилната страница.

Категория: основна

Предусловия: отваряне на профилната страница

Успешни постусловия: Визуализация на информацията за потребителя – училище, град, обща информация

Актьори: Всички аутентикирани потребители

* Случай на употреба: Регистрация на МОН потребител

Цел: Задейства се, когато се кликне създай потребител на МОН

Категория: основна

Предусловия: отваряне на админ панел

Успешни постусловия: Създаване на потребител на МОН

Актьори: Администратор

* Случай на употреба: Изтриване на потребители

Цел: Задейства се, в таблица с всички потребители

Предусловия: отваряне на админ панел

Успешни постусловия: Изтрива се конкретен потребител

Актьори: Администратор

* Случай на употреба: Одобрение на учител

Цел: Задейства се, когато се кликне одобри учител

Категория: основна

Предусловия: отваряне на админ панел

Успешни постусловия: Променя се статуса на учител на Approved

Актьори: Администратор

* Случай на употреба: Отказване на учител

Цел: Задейства се, когато се кликне откажи учител

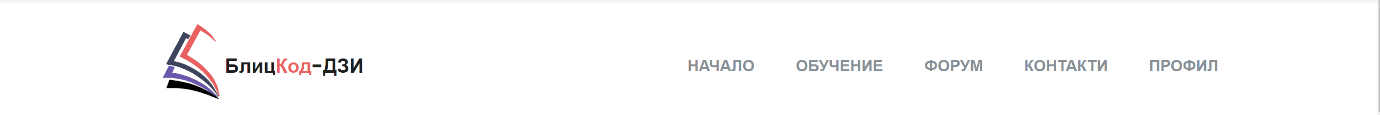
Категория: основна

Предусловия: отваряне на админ панел

Успешни постусловия: Променя се статуса на учител на Disapproved

Актьори: Администратор

**2.3. Потребителски интерфейс /менюта, екрани, Views/**

1. Главни страници – в проекта има различни изгледи в зависимост от ролята на потребителя, но при всеки случай има „главни страници“ и следствия от тях.
   1. Навигация (navigation bar)
      1. Не аутентикиран потребител
      2. Аутентикиран потребител
   2. Регистрация / Вход
      1. A screenshot of a login form

         Description automatically generatedВход (Фиг 2)

(Фиг 2)

* + 1. Регистрация
* Има два вида регистрация – за ученик и учител. При регистрацията се получава съответната роля.
* При регистрация на учител се появяват допълнителни полета
* Регистрацията на служител на МОН(MonUser) се прави от администратор, защото акаунта трябва да бъде ауторизиран
* При избора на училище, потребителите избират първо града, а след това училището. С идея по-лесна ауторизация на учителите, в базата са въведени(чрез конфигурация) всички училища в България, които имат гимназиален етап на обучение.
  + - 1. A screenshot of a computer

         Description automatically generated За ученик (Фиг 3)

(Фиг 3)

* + - 1. A screenshot of a computer

         Description automatically generated За учител (допълнителни полета) (Фиг 4)

(Фиг 4)

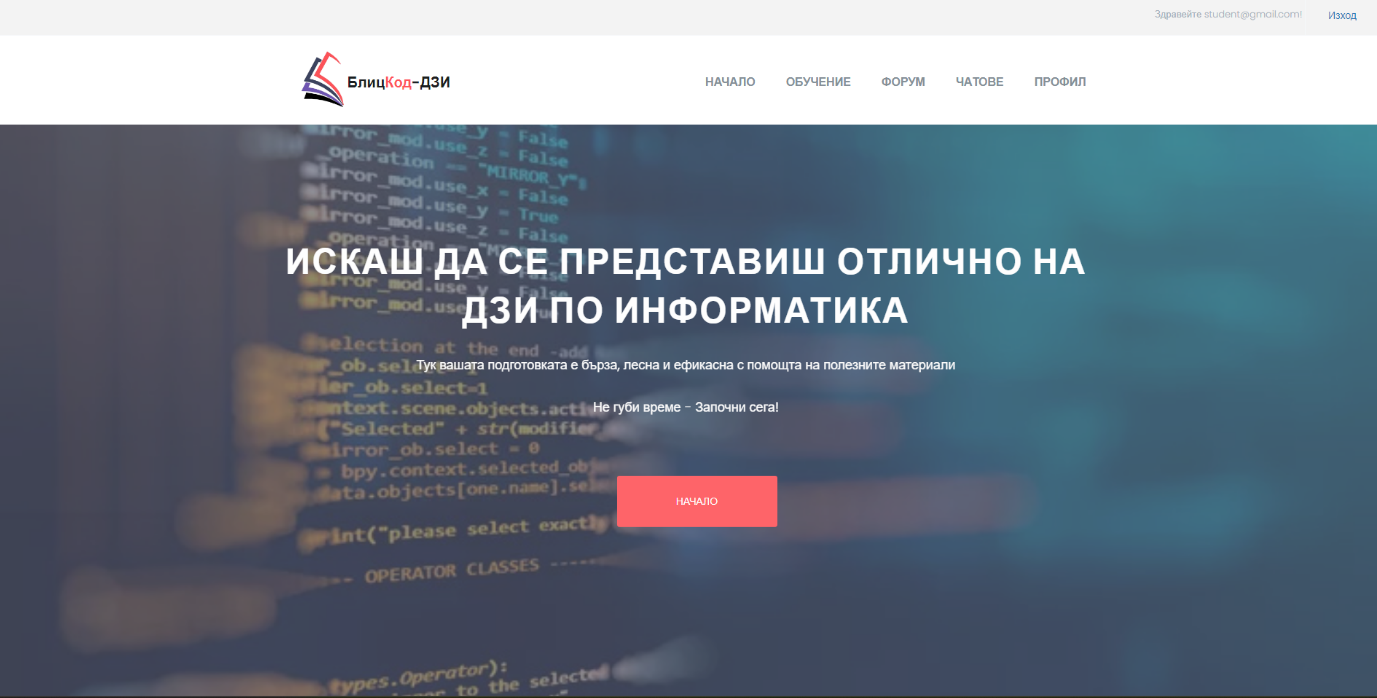
* 1. Начална страница
     1. Аутентикиран потребител (Фиг 5)
* Информация за функционалността на приложението за улеснение на потребителите. При кликане пренасочва към съответната страница.
* Началната страница съдържа и новини качени от МОН. Идеята на тази функционалност е потребителите да бъдат информирани с най-новите промени в учебната програма и изпитите.
* Също така има ревюта на клиенти. По този начин потребителите виждат истински мнения на потребители.

A screenshot of a website

Description automatically generated

(Фиг 5)

* + 1. Анонимен потребител (Фиг 6)
* Информация за ролята, с която потребител може да се регистрира и да участва в приложението.
* Също така има ревюта на клиенти. По този начин потребителите виждат истински мнения на потребители.



(Фиг 6)

A screenshot of a computer

Description automatically generated

(Фиг 6)

* 1. Обучение (Фиг 7)
* Обучението представлява систематизирано изобразяване на учебната програма. Тя е разделена на 4 модула, които се изучават в 11 и 12 клас. Всеки подул има раздели, който съдържат в себе си уроци и упражнения.
* Всичко свързано с учебната програма се променя и редактира от потребители с ролята monUser. Това е така, защото се цели главните презентации и документи, който се изобразяват в уроците и упражненията, да бъдат надеждни и валидирани.
* Обучението(системата) е абсолютно съвпадаща с учебната програма. По този начин намирането на материали и уроци е улеснено за потребителите, защото подготовката за матурата може да бъде извършена систематизирано, постепенно и последователно, като върви в крачка с учебния процес в училище.
* Цялата учебна програма е сийдната чрез конфигурации.

A screenshot of a computer

Description automatically generated

(Фиг 7)

* 1. Раздел от обучението
* Разделите, който следователно включват в себе си уроци и упражнения имат различен изглед за ученици, учители и потребител от министерството на образованието(monUser).
  + 1. За ученик и учител (Фиг 8)
* Учениците и учителите могат да достъпват уроците и упражненията като кликнат върху съответното разделение.

A screenshot of a computer

Description automatically generated

(Фиг 8)

* + 1. За представител на МОН(monUser) (Фиг 9)
* A screenshot of a computer

  Description automatically generatedБутоните добави и промени дават възможността на потребителя да добавя главни материали и информация в съответния урок или упражнение.

(Фиг 9)

* 1. Създаване на учебен материал
     1. В урок (Фиг 10)
* При създаване на урок потребителя може да добави обобщена информация, която да насочва за главите теми, който засява текущия урок. Също така задължително се добавя презентация. Направена е валидация на файловете, който могат да бъдат качени. При създаване на урок може да бъде качена само презентация.
* Когато файлът е качен той се обработва с помощта на Syncfusion и се конвертира в pdf формат. Това се прави, защото файловете на Microsoft(.pptx, docx ….) не могат да бъдат визуализирани в уеб пространството.
* Прави се заявка до облачното пространство (Cloudinary). Качването на големи файлове е бавна операция, затова има loader. Качването е оптимизирано като е асинхронно.
* Когато файлът е качен той се обработва с помощта на Syncfusion и се конвертира в pdf формат. Това се прави, защото файловете на Microsoft(.pptx, docx ….) не могат да бъдат визуализирани в уеб пространството.
* Прави се заявка до облачното пространство (Cloudinary). Качването на големи файлове е бавна операция, затова има loader. Качването е оптимизирано като е асинхронно.

A screenshot of a computer

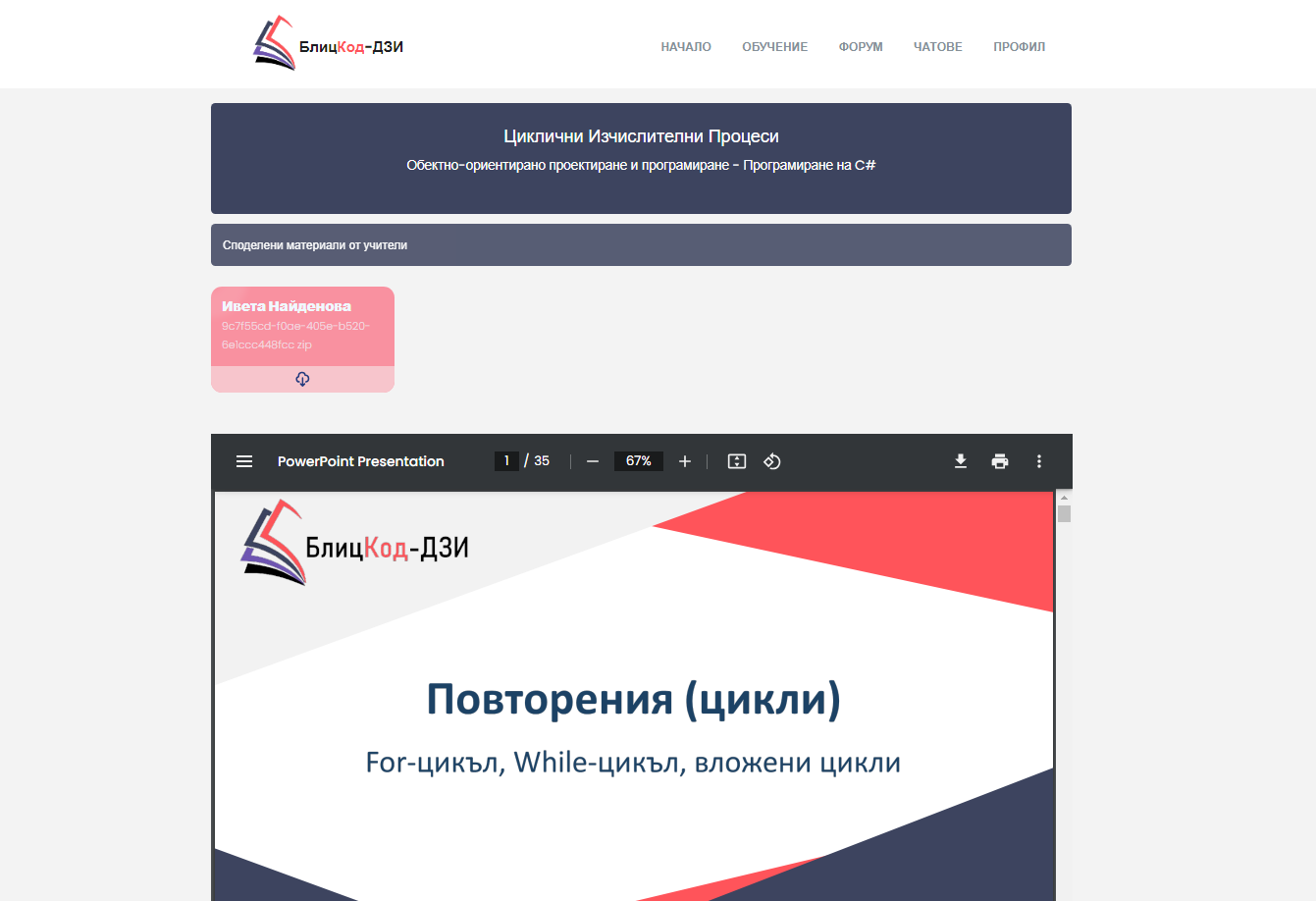
Description automatically generated

(Фиг 10)

* + 1. В упражнение: (Фиг 11)
* A screenshot of a computer

  Description automatically generatedАналогично на уроците. Разликата е, че може да се качват само документи (.docx)

(Фиг 11)

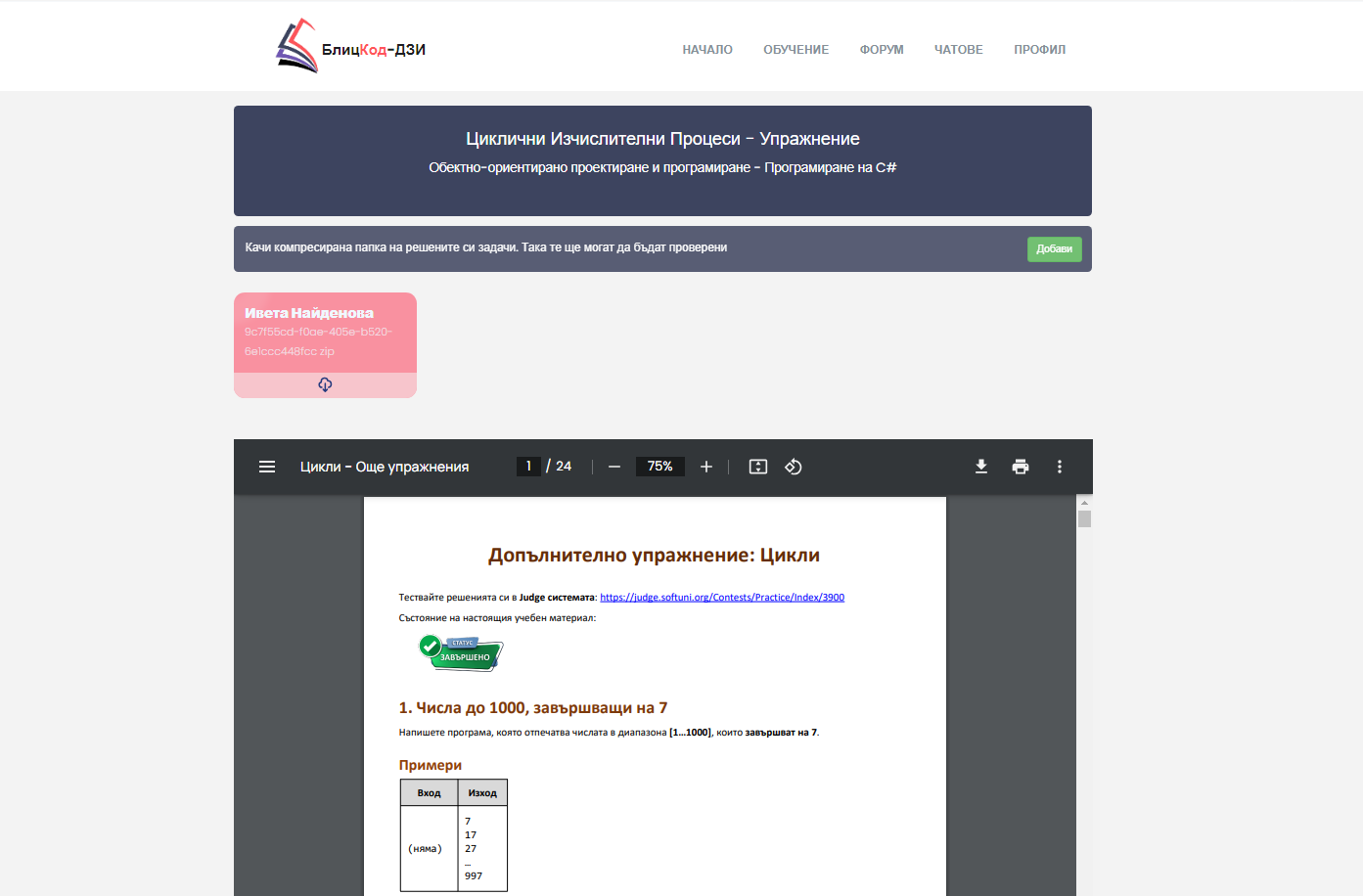
* 1. Урок
* Уроците съдържат в себе си презентация, която е визуализирана чрез iframe.
* В уроците има сложна функционалност свързана с качването и тегленето на материали, който са споделени от учители.
* Идеята е потребителите да имат възможно най-много материали, от който да учат.
* Учителите могат да качват материалите си, като това се случва отново асинхронно. Материалите трябва да бъдат компресирана папка.
  + 1. За ученик (Фиг 12)
* Учениците имат достъп до материалите на учители. Като могат да ги теглят.
* Също така потребителите могат да теглят главната презентация, която е качена от потребител на МОН.

(Фиг 12)

* + 1. За учител (Фиг 13)
* A screenshot of a computer

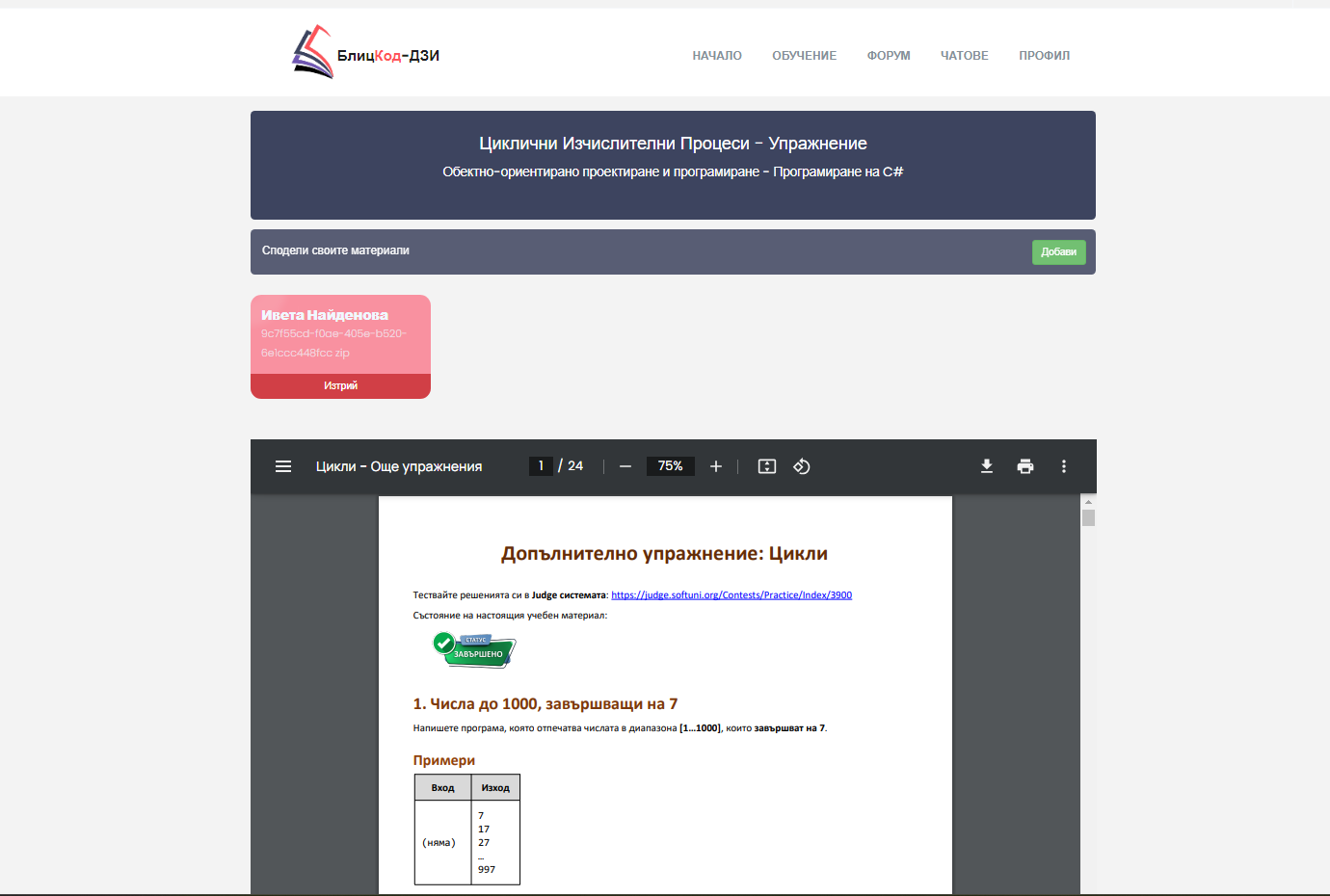
  Description automatically generatedУчителите могат да качват материалите си. Също така могат да ги трият.

(Фиг 13)

* 1. Упражнение
* Упражненията са аналогични на уроците. Разликата е, че в упражненията, качените документи съдържат задачи. Тези задачи когато бъдат решени, учениците могат да качат компресирана папка на задачите си, като от там учителите след това могат да ги проверят и да им върнат отново компресирана папка, като вътре сами могат да преценят как да поправят грешките и да оставят рецензия.
* Върнатите материали са видими в профилните страници на потребителите
  + 1. За ученик (Фиг 14)

(Фиг 14)

* + 1. За учител (Фиг 15)



(Фиг 15)

* 1. Сваляне на файлове (Фиг 16)
* В таква розова кутийка се изобразяват всички материали. С бутона на облаче съответният материал може да бъде изтеглен.
* Всички действия с всички видове файлосе се извърва със заявки към облачното пространство.
* В базата се пазят единствено пътищата към файловете.

A screenshot of a computer

Description automatically generated

(Фиг 16)

1. Новини

* Потребителите на Мон могат да качват новини. Видими са на началната страница.
* Създателя на новината може да я променя и изтрива.
  1. 1. Изглед за потребител в ролята monUser (Фиг 17)

A screenshot of a computer

Description automatically generated

(Фиг 17)

* + 1. A screenshot of a computer

       Description automatically generatedЗа всички други потребители (Фиг 18)

(Фиг 18)

* + 1. Създаване на новина и редактиране (Фиг 19)

A screenshot of a computer

Description automatically generated

(Фиг 19)

* 1. Страница за контакти (Фиг 20)
* Страницата за контакти съдържа информация за администратора. Тук е написана информацията за контакт.
* A screenshot of a computer

  Description automatically generatedСъщо така тук потребителите могат да изразят своето мнение за приложението. Така се създава ревю.

(Фиг 20)

* 1. Ревюта (Фиг 21)
* A screenshot of a computer

  Description automatically generatedВсички ревюта на потребители се визуализират в началната страница.

(Фиг 21)

1. Форум

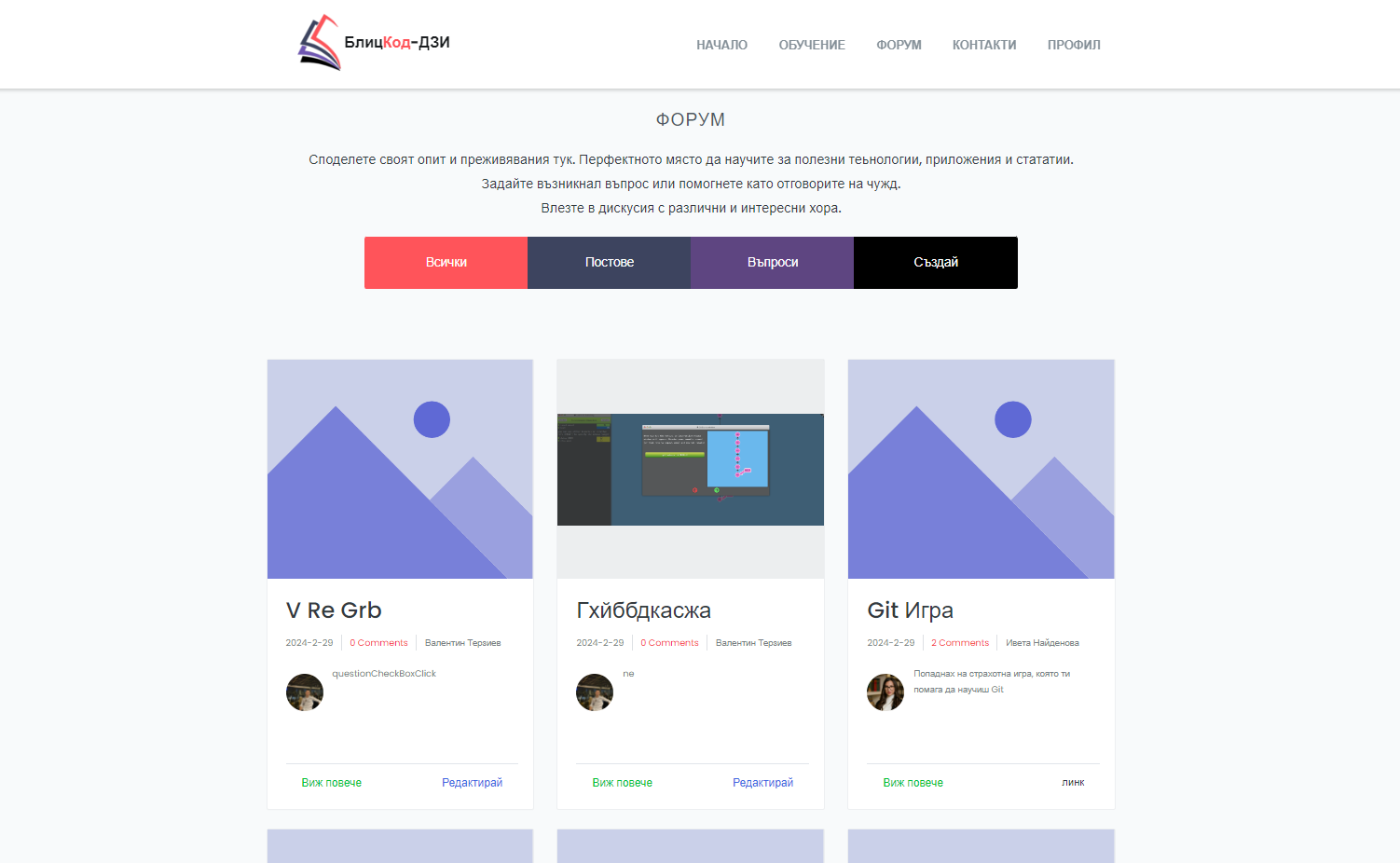
* Форумът в БлицКод е с две предназначения – потребителите да имат възможността да задават въпроси свързани със затруднения и неясности, както и да споделят интересни и полезни за учебния процес неща под формата на постове.
* Форумът има филтрация - може да бъдат визуализирани само постове(интересни неща, съвети и тн) или само зададени въпроси(затруднения), или всичко заедно, сортирано по дата.
* Под всеки пост потребителите могат да споделят мнението си, да задават въпроси или да отговарят на въпроси, под формата на коментари(секция за коментари).
* Идеята зад форума е създаване на малко, но полезно пространство(като социална мрежа, блог) за хора с еднакви интереси, а именно желанието да се самоусъвършенстват в сферата на информатика, и по конкретно информатиката, която е покрита от учебната програма.
* Форумът е видим и за анонимни потребители, но те не могат да участват в него, нито могат да достъпват детайлите.
  1. A screenshot of a computer

     Description automatically generatedАнонимен потребител (Фиг 22)

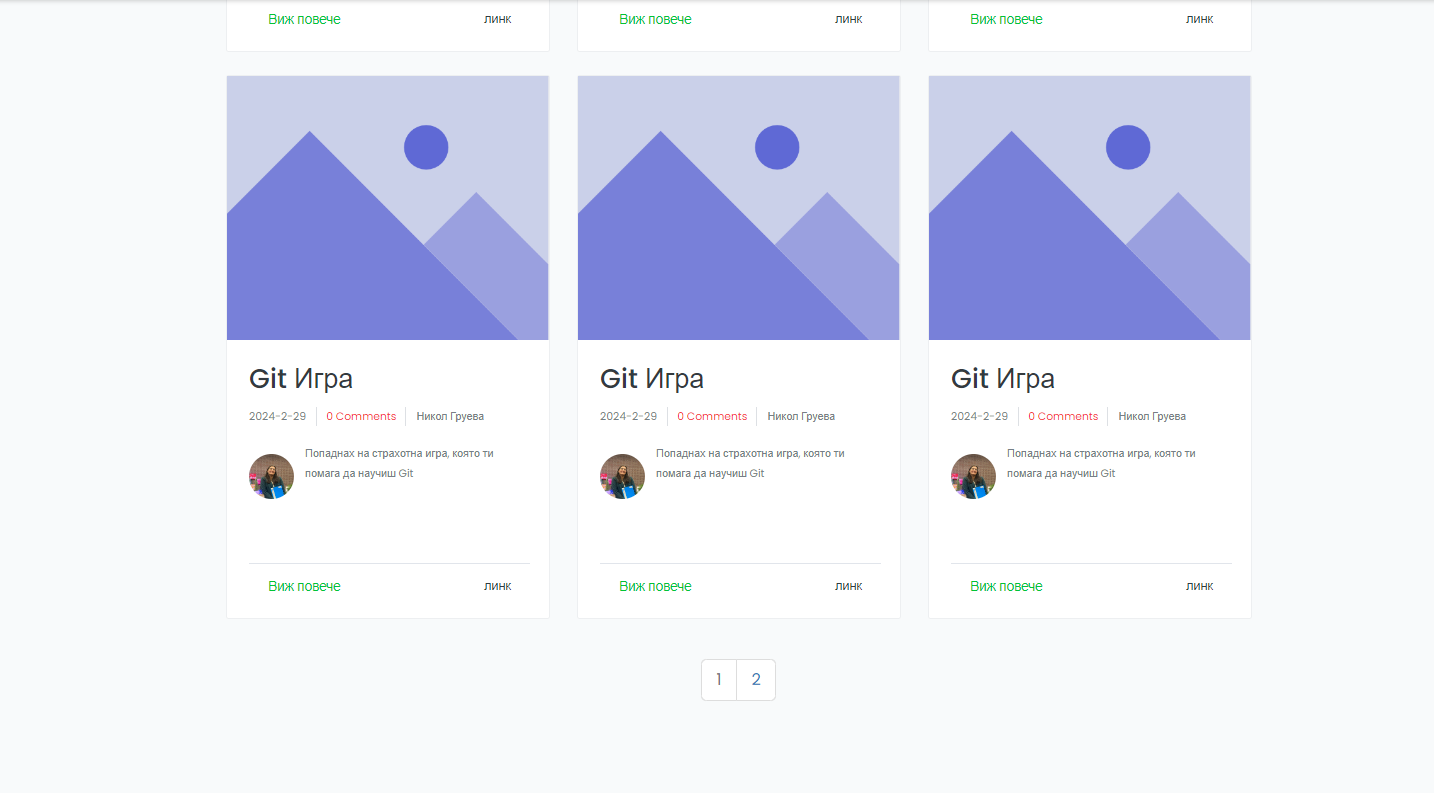
(Фиг 22)

* 1. Ауторизиран потребител (Фиг 23)

(Фиг 23)

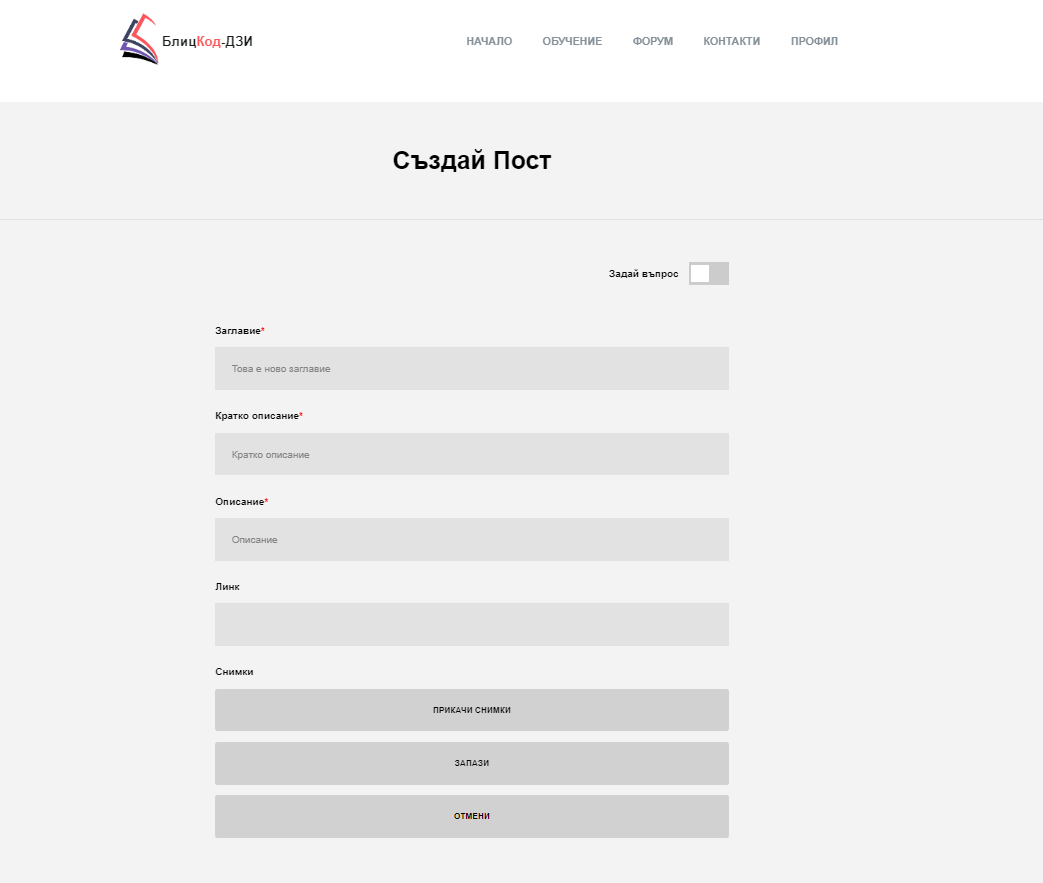


* 1. Pagination(номериране) (Фиг 24)
* Форумът се очаква да бъде обемен, т.е. да има много на брой постове и следователно коментари. Затова е имплементиран pagination(номерация). По този начин, от базата се извличат по 9 поста на страница, което облекчава приложението и подобрява потребителското преживяване, защото всичко работи бързо и плавно.
* Pagination е имплементиран и в детайлната страница на всеки пост, където има коментари.



(Фиг 24)

* 1. Създаване, редактиране и изтриване на пост (Фиг 25)
* Редактирането и изтриването на постове може да се случи чрез бутоните в страницата форум или в профилната страница на съответния потребител.



(Фиг 25)

A screenshot of a computer

Description automatically generated

(Фиг 25)

* 1. Детайлна страница на пост (Фиг 26)
* A screenshot of a computer

  Description automatically generatedВ детайлната страница на пост има цялата информация за поста. Ако има качени снимки те се изобразяват в каросел. Под информацията на поста е позиционирана секцията за коментари

(Фиг 26)

* 1. Коментари (Фиг 27)
* В един коментар може да има както информация, така и снимки. Това дава свобода на потребителя да изразява мнението си по-лесно, свободно и точно.
* A screenshot of a computer

  Description automatically generatedСекцията за коментари отново има pagination

(Фиг 27)

1. Профилна страница (Фиг 28)

* Профилната страница съдържа цялата информация за потребителя: Всички качени постове, зададени въпроси, качени материали. Ако потребителят е учител има информация и за всички задачи, който е проверил и изпратил на ученик. Аналогично за ученик има информация за всички задачи, който са му върнати, т.е. проверени.
* В профилната страница има връзки към съответните материали, постове, въпроси. Също така има бутони за редактиране и триене на вече съществуващите неща

A screenshot of a computer

Description automatically generated

(Фиг 28)

1. Администратор
   1. Регистрация на МОН потребител (Фиг 29)

* A white rectangular object with a blue background

  Description automatically generated with medium confidenceАдминистратора може да регистрира потребители на МОН. Те не могат да се регистрират сами, защото трябва да бъде валидирана тяхната самоличност

(Фиг 29)

* 1. Триене на потребители (Фиг 30)
* Триенето на потребители се извършва от администратора. Те не се изтриват, а само се деактивират, защото триенето от база от данни е лоша практика.
* В таблицата с потребители има филтрация по техните роли

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

(Фиг 30)

* 1. Одобрение и отказване на учители (Фиг 31)
* A close-up of a computer screen

  Description automatically generatedАдминистратора може да одобрява и отхвърля учители. При одобрение те получават роля учител и съответните права към нея. При отхвърляне статуса се променя от Waiting в Disapproved.

(Фиг 31)

**2.4.** **Избрани технологии и софтуерните средства за разработка на приложението:**

* + 1. **Visual Studio**

Visual Studio e IDE, в което може да се разработва софтуер на C#. Дава възможност да се създават, обвързват и програмират обекти.

* + 1. **Microsoft SQL Server** е система за управление на релационни бази данни, разработена от Microsoft.
    2. **SSMS – Sql Server Management Studio**

В SSMS се визуализира базата на приложението.

* + 1. **C# и ASP.NET Core** **6.0**

C# е стандартен силно типизиран език за програмиране. ASP.NET Core e framework за изготвяне на уеб приложения. Използвано е в приложението за цялостната разработка и осъществяване на MVC структурата.

* + 1. **Entity Framework**

Entity Framework помага за създаване на базата по метода Code First.

* + 1. **JavaScript**

JavaScript е интерпретируем скриптов език от високо ниво, който позволява на разработчиците да създават интерактивни уебстраници. JavaScript се използва за добавяне на специални ефекти към страниците и за показване на данни по интерактивен начин.

* + 1. **Bootstrap 5**

Bootstrap е безплатен фреймуърк с отворен код и служи за създаване на интерфейсите на уеб приложения. Това е най-популярната HTML, CSS и JS библиотека за разработване на адаптивни, предназначени на първо място за мобилни устройства уеб интерфейси. Сайтовете изградени с Bootstrap притежават responsive дизайн, това означава че автоматично се мащабират за различните устройствата – независимо дали устройството е мобилен телефон, таблет, лаптоп, настолен компютър, екранен четец и т.н. Използван е в проекта за изработка на VIEWS

* + 1. **HTML 5** ни помага да зададем основната структура на уеб страниците. Полезен е за редактиране на параграфи, заглавия и други основни елементи на уеб страница.
    2. **CSS 3-** език за описание на стилове, които указват как елементите в една Интернет страница ще бъдат форматирани. С негова помощ стилизираме едновременно всички страници от сайта. Чрез CSS се отделя описанието на съдържанието и структурата на сайтовете от стилизирането им. Всички браузъри поддържат CSS.
    3. **Cloudinary**

API за съхранение на файлове. Там се съхраняват снимките от проекта.

* + 1. **Syncfusion**

Nuget package (библиотека) за обработка на файлове. Сменям формата на презентации и документи от съответния формат в .pdf.

**III.** **ГЛАВА - ПРОГРАМНА РЕАЛИЗАЦИЯ**

A computer screen shot of a computer

Description automatically generated**3.1.** **Структурата на базата от данни(Фиг 32)**

(Фиг 32)

Основната таблица е AspNetUsers, към която има връзка от таблицата AspNetUserLogins чрез полето UserId. UserId се явява foreign key в таблица AspNetUserLogins и така се реализира връзка 1:много. Таблиците се създават от Identity системата, като в основната таблица има допълнително добавени полета (*FirstName, LastName, Cit, About me, CreatedOn, RoleName, ProfilePictureUrl, IsActive, Posts, Materials, Reviews).* Таблицата AspNetUsers се свързва и с таблицата Posts с полето Posts и връзката е много:1. Таблицата AspNetUsers се свързва и с таблицата Reviews с полето Reviews и връзката е много:1. Таблицата AspNetUsers се свързва и с таблицата Materials с полето Materials и връзката е много:1.

Tаблицата AspNetUserRoles се свързва с таблицата AspNetUsers чрез полето UserId. Таблица AspNetUserRoles се свързва и към таблицата AspNetRoles чрез полето RoleId и така между таблиците AspNetUsers и AspNetRoles се реализира връзка много: много.

Таблицата Teachers се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и връзката е 1:1. Таблицата Teacher се свързва с таблицата School с полето SchoolId, което се явява foreign key и връзката е 1:много. Таблицата Teacher се свързва с таблицата Lesson с полето Lessons, което се явява foreign key към mapping таблицата LessonTeacher, която реализира връзка много:много. Таблицата Teacher се свързва с таблицата Exercise с полето Exercises, което се явява foreign key към mapping таблицата ExerciseTeacher, която реализира връзка много:много.

Таблицата Students се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и връзката е 1:1. Таблицата Students се свързва с таблицата School с полето SchoolId, което се явява foreign key и връзката е 1:много. Таблицата Students се свързва с таблицата Exercise с полето FinishedExercises, което се явява foreign key към mapping таблицата ExerciseStudent, която реализира връзка много:много.

Таблицата SectionOfCurricular се свързва с таблицата ModuleOfCurricular с полето ModuleOfCurricularId, което се явява foreign key и образува връзка 1:много. Таблицата SectionOfCurricular се свързва с таблицата Lessons с полето Lessons, което се явява foreign key и образува връзка много:1. Таблицата SectionOfCurricular се свързва с таблицата Exercises с полето Exercises, което се явява foreign key и образува връзка много:1.

Таблицата Schools се свързва с таблицата Students с полето Students, което се явява foreign key и образува връзка 1:много. Таблицата Schools се cвързва с таблицата Teachers с полето Teachers, което се явява foreign key и образува връзка 1:много.

Таблицата Reviews се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и образува връзка много:1.

Таблицата Posts се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и образува връзка много:1. Таблицата Posts се cвързва с таблицата Comments с полето Comments, което се явява foreign key и образува връзка 1:много. Таблицата Posts се cвързва с таблицата Images с полето Images, което се явява foreign key и образува връзка 1:много.

Таблицата News се свързва с таблицата MonUsers с полето MonUserId, което се явява foreign key и образува връзка много:1. Таблицата News се cвързва с таблицата Images с полето Images, което се явява foreign key и образува връзка 1:много.

Таблицата MonUsers се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и връзката е 1:1. Таблицата MonUser се свързва с таблицата News с полето News, което се явява foreign key и връзката е 1:много. Таблицата MonUser се свързва с таблицата Exercise с полето Lesson, което се явява foreign key към mapping таблицата LessonMonUser, която реализира връзка много:много.

Таблицата MоduleOfCurricular се свързва с таблицата SectionOfCurricular с полето SectionOfCurricularId, което се явява foreign key и образува връзка много:1.

Таблицата Lessons се свързва с таблица SectionOfCurricular с полето SectionOfCurricularId, което се явява foreign key и образува връзка много:1. Таблицата Lessons се свързва с таблица Exercises с полето ExersiceId, което се явява foreign key и образува връзка 1:1. Таблицата Lessons се свързва с таблица LessonMonUser с полето LessonMonUser, което се явява foreign key и образува връзка много:много. Таблицата Lessons се свързва с таблица Teachers с полето Teachers, което се явява foreign key и образува връзка много:много. Таблицата Lessons се свързва с таблица LessonMaterials с полето LessonMaterials, което се явява foreign key и образува връзка много:много.

Таблицата ImageComments се свързва с таблицата Comments полето CommentId, което се явява foreign key и образува връзка много:1.

Таблицата Image се свързва с таблицата Posts полето Postd, което се явява foreign key и образува връзка много:1.

Таблицата LessonMaterial се свързва с таблицата AspNetUsers полето UserId, което се явява foreign key и образува връзка много:1. Таблицата LessonMaterial се свързва с таблицата Lesson полето LessonId, което се явява foreign key и образува връзка много:1.

Таблицата ExerciseMaterial се свързва с таблицата AspNetUsers полето UserId, което се явява foreign key и образува връзка много:1. Таблицата ExerciseMaterial се свързва с таблицата Exercise полето ExerciseId, което се явява foreign key и образува връзка много:1.

Таблицата Exercise се свързва с таблица SectionOfCurricular с полето SectionOfCurricularId, което се явява foreign key и образува връзка много:1. Таблицата Lessons се свързва с таблица Exercises с полето ExersiceId, което се явява foreign key и образува връзка 1:1. Таблицата Exercise се свързва с таблица Students с полето Students, което се явява foreign key и образува връзка много:много. Таблицата Lessons се свързва с таблица ExrciseMaterials с полето ExrciseMaterials, което се явява foreign key и образува връзка много:много.

Таблицата Comment се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и образува връзка много:1. Таблицата Comment се свързва с таблицата Post с полето PostId, което се явява foreign key и образува връзка много:1. Таблицата Comment се свързва с таблицата Images с полето Images, което се явява foreign key и образува връзка 1:много.

Таблицата Answer се свързва с таблицата AspNetUsers с полето UserId, което се явява foreign key и образува връзка много:1. Таблицата Answer се свързва с таблицата ExerciseMaterial с полето ExerciseMaterial, което се явява foreign key и образува връзка много:1.

**3.2.** **Структура на MVC приложението**

MVC е съкратено от Model-view-controller. Представлява архитектурен

шаблон при програмния дизайн. Той отговаря за разделянето на бизнес логиката на три взаимосвързани части: Model, View и Controler.

* Model –моделите пазят всички данни, с които искаме да работим;
* View – интерфейсът на приложението, който визуализира наличните, обработени данни;
* Controller – съдържат бизнес логиката, която е описана в методи на сървиси, които се извикват в контролера и така обработените данни се подават към изгледите. Контролерът действа като посредник – той комбинира модела с изглед и предоставя резултат на крайния потребител

В това приложение архитектурата отново е MVC, но е разделена в отделни приложения.

Проектът е съставен от 4 подпроекта:

1. **PreparationForITExam** – включва views, controllers, admin area, extension classes

2. **PreparationForITExam.Core** – константи, интерфейси, пояснителни текстове при грешки, view models и models, services (функционалността на проекта, чрез сървисите се постига по-добра абстрактност и зависимости)

3. **PreparationForITExam.Infrastructure** – entities за базата данни, константи за валидация на полетата на базата, dbContext, миграции, конфигурации (за сийдване на базата)

4. **PreparationForITExam.Tests –** съдържа unit tests, които тестват 62% от бизнес логиката с 84 теста

**3.2.1. Описание на CRUD операциите**

В разработеното приложение има два екшън метода **Create**, които служат за изпълнение на първата от четирите CRUD операции – Create.

Първият метод се използва за извикване на Get заявка.

[HttpGet]

public async Task<IActionResult> Add()

{

var model = new NewsFormViewModel();

model.UserId = this.User.Id();

return View(model);

}

Този метод връща изглед:

@model NewsFormViewModel

<section>

<div class="row cont">

<div class="col-sm-12 offset-lg-2 col-lg-8 offset-xl-3 col-xl-6">

<**form** method ="post" enctype="multipart/form-data">

<**input** **type**="hidden" **asp-for**="@Model.UserId" />

<**input** **type**="hidden" **asp-for**

">Съдържание<span class="mandatory">\*</span></label>

<**input** **asp-for**="Description" class="h-full-width" **type**="text" placeholder="Днешните изпити ..." required>

<**span** **asp-validation-for**="Description" class="text-danger"></**span**>

</div>="@Model.Id" />

<div>

<label for="sampleInput">Заглавие<span class="mandatory">\*</span></label>

<**input** **asp-for**="Title" class="h-full-width" **type**="text" placeholder="Това е ново заглавие" required>

<**span** **asp-validation-for**="Title" class="text-danger"></**span**>

</div>

<div>

<label for="sampleInput

<div>

<label for="sampleInput">Линк<span class="mandatory">\*</span></label>

<**input** **asp-for**="UsefulUrls" class="h-full-width" **type**="text" placeholder="https://web.mon.bg/bg/101234" required>

<**span** **asp-validation-for**="UsefulUrls" class="text-danger"></**span**>

</div>

<div class="mb-3">

<input class="btn h-full-width btn-green-h" type="submit" value="Запази">

<**a** **asp-controller**="Home" **asp-action**="Index" class="btn h-full-width">Отмени</**a**>

</div>

</**form**>

</div>

</div>

</section>

@model NewsFormViewModel

@{

ViewBag.Title = "Създай новина";

}

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<section>

<h2 class="text-center">@ViewBag.Title</h2>

<hr />

<**partial** **name**="\_NewsFormPartial" **model**="@Model" />

</section>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

Вторият Create метод се използва за извикване на Post заявка. При извикването на този метод заявката взима попълнените във формата данни, праща ги на сървъра и чрез тях създава нов запис в базата данни.

[HttpPost]

public async Task<IActionResult> Add(NewsFormViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

await newsService.AddNews(model);

return RedirectToAction("Index", "Home");

}

Този метод използва бизнес логика от newsService

public async Task AddNews(NewsFormViewModel model)

{

var news = new News();

news.Title = model.Title;

news.Description = model.Description;

news.UsefulUrls = model.UsefulUrls;

news.MonId = await monUserService.GetMonUserIdByUserId(model.UserId);

await repo.AddAsync(news);

await repo.SaveChangesAsync();

}

В разработеното приложение има два екшън метода **GetOnlyPosts**, които служат за изпълнение на първата от четирите CRUD операции – Read.

Първият метод се използва за извикване на Get заявка.

[HttpGet]

public async Task<IActionResult> GetOnlyPosts(int id)

{

var model = await postService.GetOnlyPosts(id);

var count = await postService.GetOnlyPostsCount();

TempData["pages"] = count % 9 == 0 ? count / 9 : (count / 9) + 1; ;

TempData["curr"] = id == 0 ? 1 : id;

return View("Index", model);

}

Този метод връща изглед:

@model List<PostViewModel>

@{

}

<section id="explore" class="explore">

<div class="container">

<div class="section-header">

<h2>Форум</h2>

<p>

Споделете своят опит и преживявания тук. Перфектното място да научите за полезни теьнологии, приложения и стататии. <br>

Задайте възникнал въпрос или помогнете като отговорите на чужд. <br>

Влезте в дискусия с различни и интересни хора.

</p>

<br>

<div class="btn-group" role="group">

<**a** class="btn welcome-hero-btn" **asp-controller**="Post" **asp-action**="Index">

Всички

</**a**>

<**a** style="background:#3D445F!important" class="btn welcome-hero-btn" **asp-controller**="Post" **asp-action**="GetOnlyPosts">

Постове

</**a**>

<**a** style="background:#5e4581!important" class="btn welcome-hero-btn" **asp-controller**="Post" **asp-action**="GetOnlyQuestions">

Въпроси

</**a**>

@if (this.User.Identity.IsAuthenticated)

{

<**a** style="background:#000!important" class="btn welcome-hero-btn" **asp-controller**="Post" **asp-action**="Add">

Създай

</**a**>

}

</div>

</div><!--/.section-header-->

<div class="explore-content">

<div class="row">

@foreach (var post in Model)

{

<div class=" col-md-4 col-sm-6">

<div class="single-explore-item">

@if (post.Images.Count == 0)

{

<div class="single-explore-img">

<**img** src="~/images/icons/gallery.png" alt="explore image">

</div>

}

else

{

<div class="single-explore-img">

<img src="@post.Images[0]" alt="explore image">

</div>

}

<div class="single-explore-txt bg-theme-1">

<h2><a href="#">@post.Title</a></h2>

<p class="explore-rating-price">

<a href="#"> @post.PostedOn</a>

<span class="explore-price-box">

<span class="explore-price">@post.CommentsCount comments</span>

</span>

<**a** **asp-controller**="Profile" **asp-action**="Index" **asp-route-id**="@post.UserId">@post.UserName</**a**>

</p>

<div class="explore-person">

<div class="row">

<div class="col-sm-2">

<**a** **asp-controller**="Profile" **asp-action**="Index" **asp-route-id**="@post.UserId">

<div class="testimonial-img">

<img src="@post.UserProfilePicture" alt="explore person">

</div>

</**a**>

</div>

<div class="col-sm-10">

<p style="margin-left: 0.5em;">

@post.ShortDescription

</p>

</div>

</div>

</div>

@if (this.User.Identity.IsAuthenticated)

{

<div class="explore-open-close-part">

<div class="row">

<div class="col-sm-5">

<**a** class="close-btn open-btn" **asp-controller**="Post" **asp-action**="Details" **asp-route-id**="@post.Id">Виж повече</**a**>

</div>

<div class="col-sm-7">

<div class="explore-map-icon">

@if (this.User.Id() == post.UserId && post.UsefulUrl != null)

{

<**a** class="close-btn" **asp-controller**="Post" **asp-action**="Edit" **asp-route-id**="@post.Id">Редактирай</**a**>

}

else if (this.User.Id() != post.UserId && post.UsefulUrl != null)

{

<a href="@post.UsefulUrl">линк</a>

}

else if (this.User.Id() == post.UserId && post.UsefulUrl == null)

{

<**a** class="close-btn" **asp-controller**="Post" **asp-action**="Edit" **asp-route-id**="@post.Id">Редактирай</**a**>

}

</div>

</div>

</div>

</div>

}

</div>

</div>

</div>

}

</div>

</div>

<nav style="display: ruby-text;">

<ul class="pagination pagination-lg">

@for (int i = 1; i <= (int)TempData["pages"]; i++)

{

if ($"{i}" == TempData["curr"].ToString())

{

<li class="page-item disabled">

<a class="page-link" href="#" tabindex="-1">@i</a>

</li>

}

else

{

<li class="page-item">

<**a** class="page-link" **asp-controller**="Post" **asp-action**="Index" **asp-route-id**="@i">@i</**a**>

</li>

}

}

</ul>

</nav>

</div>

<!--/.container-->

</section>

#

<div class="single-explore-txt bg-theme-1">

<h2><a href="#">@post.Title</a></h2>

<p class="explore-rating-price">

<a href="#"> @post.PostedOn</a>

<span class="explore-price-box">

<span class="explore-price">@post.CommentsCount comments</span>

</span>

<**a** **asp-controller**="Profile" **asp-action**="Index" **asp-route-id**="@post.UserId">@post.UserName</**a**>

</p>

<div class="explore-person">

<div class="row">

<div class="col-sm-2">

<**a** **asp-controller**="Profile" **asp-action**="Index" **asp-route-id**="@post.UserId">

<div class="testimonial-img">

<img src="@post.UserProfilePicture" alt="explore person">

</div>

</**a**>

</div>

<div class="col-sm-10">

<p style="margin-left: 0.5em;">

@post.ShortDescription

</p>

</div>

</div>

</div>

@if (this.User.Identity.IsAuthenticated)

{

<div class="explore-open-close-part">

<div class="row">

<div class="col-sm-5">

<**a** class="close-btn open-btn" **asp-controller**="Post" **asp-action**="Details" **asp-route-id**="@post.Id">Виж повече</**a**>

</div>

<div class="col-sm-7">

<div class="explore-map-icon">

@if (this.User.Id() == post.UserId && post.UsefulUrl != null)

{

<**a** class="close-btn" **asp-controller**="Post" **asp-action**="Edit" **asp-route-id**="@post.Id">Редактирай</**a**>

}

else if (this.User.Id() != post.UserId && post.UsefulUrl != null)

{

<a href="@post.UsefulUrl">линк</a>

}

else if (this.User.Id() == post.UserId && post.UsefulUrl == null)

{

<**a** class="close-btn" **asp-controller**="Post" **asp-action**="Edit" **asp-route-id**="@post.Id">Редактирай</**a**>

}

</div>

</div>

</div>

</div>

}

</div>

</div>

</div>

}

</div>

</div>

<nav style="display: ruby-text;">

<ul class="pagination pagination-lg">

@for (int i = 1; i <= (int)TempData["pages"]; i++)

{

if ($"{i}" == TempData["curr"].ToString())

{

<li class="page-item disabled">

<a class="page-link" href="#" tabindex="-1">@i</a>

</li>

}

else

{

<li class="page-item">

<**a** class="page-link" **asp-controller**="Post" **asp-action**="Index" **asp-route-id**="@i">@i</**a**>

</li>

}

}

</ul>

</nav>

</div>

<!--/.container-->

</section>

Този метод използва бизнес логика от postService

public async Task<List<PostViewModel>> GetOnlyPosts(int page)

{

var model = await repo.AllReadonly<Post>()

.Where(p => p.IsActive && p.IsItQuestion == false)

.OrderByDescending(p => p.Id)

.Skip(9 \* ((int)(page == 0 ? 1 : page) - 1))

.Take(9)

.Select(p => new PostViewModel

{

Id = p.Id,

Title = p.Title,

Description = p.Description,

ShortDescription = p.ShortDescription,

UserId = p.UserId,

UserName = p.User.FirstName + " " + p.User.LastName,

UserProfilePicture = p.User.ProfilePictureUrl,

UsefulUrl = p.UsefulUrl,

PostedOn = p.PostedOn.ToString("yyyy-M-d"),

CommentsCount = p.Comments.Where(c => c.IsActive).ToList().Count

})

.ToListAsync();

for (int i = 0; i < model.Count; i++)

{

if (model[i].Images != null)

{

model[i].Images = await imageService.GetPostImages(model[i].Id);

}

}

return model;

}

В разработеното приложение има два екшън метода **Edit**, които служат за изпълнение на първата от четирите CRUD операции – Update.

Първият метод се използва за извикване на Get заявка.

[HttpGet]

public async Task<IActionResult> Edit(int id)

{

var model = await postService.GetPostInfo(id);

return View(model);

}

Този метод връща изглед

@model PostFormViewModel

@{

ViewBag.Title = "Редактирай пост";

}

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<section>

<h2 class="text-center">@ViewBag.Title</h2>

<hr />

<**partial** **name**="\_PostFormPartial" **model**="@Model" />

</section>

@section Scripts {

<**partial** **name**="\_ValidationScriptsPartial" />

}

@model PostFormViewModel

<section>

<div class="row cont">

<div class="col-sm-12 offset-lg-2 col-lg-8 offset-xl-3 col-xl-6">

@if (ViewBag.Title == "Създай пост")

{

<div class="register-heading-container">

<h3 class="h-add-bottom"></h3>

<div class="game-developer-slider-container h-add-bottom ">

<label for="sampleInput" class="game-developer-label">Задай въпрос</label>

<label class="switch">

<input type="checkbox" id="questionCheckBox">

<span class="slider"></span>

</label>

</div>

</div>

}

<**form** method="post" enctype="multipart/form-data">

<**input** **type**="hidden" **asp-for**="@Model.UserId" />

<div>

<label for="sampleInput">Заглавие<span class="mandatory">\*</span></label>

<**input** **asp-for**="Title" class="h-full-width" **type**="text" placeholder="Това е ново заглавие" required>

<**span** **asp-validation-for**="Title" class="text-danger"></**span**>

</div>

<div>

<label for="sampleInput">Кратко описание<

<div>

<label for="sampleInput">Описание<span class="mandatory">\*</span></label>

<**input** **asp-for**="Description" class="h-full-width" **type**="text" placeholder="Описание" required>

<**span** **asp-validation-for**="Description" class="text-danger"></**span**>

</div>

<div>

<label for="sampleInput">Линк</label>

<**input** **asp-for**="UsefulUrl" class="h-full-width" **type**="text" placeholder="@Model.UsefulUrl">

<**span** **asp-validation-for**="UsefulUrl" class="text-danger"></**span**>

</div>

<div class="mb-3">

@if (ViewBag.Title == "Редактирай пост")

{

<**input** **type**="hidden" **asp-for**="@Model.IsItQuestion" />

<**input** style="display:block;" id="add" **asp-controller**="Post" **asp-action**="Edit" **asp-route-id**="@Model.Id" class="btn h-full-width btn-green-h" type="submit" value="Запази">

<**input** style="display:block;" id="add" **asp-controller**="Post" **asp-action**="Delete" **asp-route-id**="@Model.Id" class="btn h-full-width btn-red-h" type="submit" value="Изтрий">

}

else

{

<div class="mb-3">

<label class="profilePictureLabel">Снимки</label>

<label class="btn h-full-width" for="uploadProfilePicture">Прикачи Снимки</label>

<**input** **asp-for**="Images" id="uploadProfilePicture" class="upload-photo" accept="image/gif, image/jpeg, image/png">

<**span** **asp-validation-for**="Images" class="text-danger"></**span**>

</div>

<**input** style="display:block;" id="add" **asp-controller**="Post" **asp-action**="Add" class="btn h-full-width btn-green-h" type="submit" value="Запази">

<**input** style="display:none;" id="addQuestion" **asp-controller**="Post" **asp-action**="AddQuestion" class="btn h-full-width btn-green-h" type="submit" value="Запази">

}

<**a** **asp-controller**="Post" **asp-action**="Index" class="btn h-full-width">Отмени</**a**>

<div id="loading" hidden>

<**img** id="loading-image" src="~/loader.gif" alt="Loading..." />

</div>

</div>

</**form**>

</div>

</div>

</section>

<script>

$(window).on('load', function () {

$("#loading").hide();

});

$("#add").click(function () {

$("#loading").show();

});

$(window).on('load', function () {

$("#loading").hide();

});

$("#addQuestion").click(function () {

$("#loading").show();

});

</script>

<style>

#loading {

position: fixed;

display: block;

width: 100%;

height: 100%;

top: 0;

left: 0;

text-align: center;

opacity: 0.7;

background-color: #fff;

z-index: 99;

}

#loading-image {

position: absolute;

width: 300px;

z-index: 100;

top: 40%;

margin: 0;

right: 40%;

}

</style>

<script>

$(window).on('load', function () {

$("#loading").hide();

});

$("#add").click(function () {

$("#loading").show();

});

$(window).on('load', function () {

$("#loading").hide();

});

$("#addQuestion").click(function () {

$("#loading").show();

});

</script>

<style>

#loading {

position: fixed;

display: block;

width: 100%;

height: 100%;

top: 0;

left: 0;

text-align: center;

opacity: 0.7;

background-color: #fff;

z-index: 99;

}

#loading-image {

position: absolute;

width: 300px;

z-index: 100;

top: 40%;

margin: 0;

right: 40%;

}

</style>

Този метод използва бизнес логика от postService

public async Task<PostFormViewModel> GetPostInfo(int id)

{

var model = await repo.AllReadonly<Post>()

.Where(p => p.IsActive && p.Id == id)

.Select(p => new PostFormViewModel

{

Id = p.Id,

Title = p.Title,

Description = p.Description,

ShortDescription = p.ShortDescription,

UserId = p.UserId,

UsefulUrl = p.UsefulUrl,

IsItQuestion = p.IsItQuestion

})

.FirstOrDefaultAsync();

//model.Images = await imageService.GetPostImages(id);

return model;

}

Вторият Edit метод се използва за извикване на Post заявка. При извикването на този метод заявката взима попълнените във формата данни, праща ги на сървъра и чрез тях създава нов запис в базата данни.

[HttpPost]

public async Task<IActionResult> Edit(PostFormViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

await postService.Edit(model);

return RedirectToAction("Index", "Post");

}

Този метод използва бизнес логика от postService

public async Task Edit(PostFormViewModel model)

{

var post = await repo.GetByIdAsync<Post>(model.Id);

post.Title = model.Title;

post.Description = model.Description;

post.ShortDescription = model.ShortDescription;

post.UsefulUrl = model.UsefulUrl;

post.UserId = model.UserId;

post.IsItQuestion = model.IsItQuestion;

await repo.SaveChangesAsync();

}

В разработеното приложение има два екшън метода **DeletePost**, които служат за изпълнение на първата от четирите CRUD операции – Delete.

Първият метод се използва за извикване на Get заявка.

[HttpGet]

public IActionResult Delete(int id)

{

var model = new PostViewModel

{

Id = id,

};

return View(model);

}

Този метод връща изглед

@model PostViewModel

<**link** rel="stylesheet" **href**="~/css/site.css" **asp-append-version**="true" />

<section>

<h2 style="text-transform:none!important;" class="text-center">Сигурни ли сте, че искате да изтриете вашият пост?</h2>

<hr />

<div class="row cont">

<div class="col-sm-12 offset-lg-2 col-lg-8 offset-xl-3 col-xl-6">

<**form** method="post" enctype="multipart/form-data">

<**input** **type**="hidden" **asp-for**="@Model.Id" />

<div class="mb-3">

<input class="btn h-full-width btn-red-h" type="submit" value="Изтрий">

<**a** **asp-controller**="Post" **asp-action**="Details" **asp-route-id**="@Model.Id" class="btn h-full-width">Отмени</**a**>

</div>

</**form**>

</div>

</div>

</section>

Вторият Delete метод се използва за извикване на Post заявка. При извикването на този метод заявката взима попълнените във формата данни, праща ги на сървъра и чрез тях създава нов запис в базата данни.

[HttpPost]

public async Task<IActionResult> Delete(PostViewModel model)

{

await postService.Delete(model.Id);

return RedirectToAction("Index", "Post");

}

Методът използва бизнес логика от postService

public async Task Delete(int id)

{

var post = await repo.GetByIdAsync<Post>(id);

if (post == null)

{

throw new NullReferenceException(GlobalExceptions.PostDoesNotExistExceptionMessage);

}

post.IsActive = false;

await repo.SaveChangesAsync();

}

**3.2.2. Филтриране**

- В администраторския панел има филтрация на потребителите по тяхната роля

[HttpPost]

public async Task<IActionResult> Index(AdminViewModel model)

{

var filter = model.Filtration;

if (filter == 0) // All

{

model.Users = await userService.GetAllUsers(0);

var count = await userService.GetAllUsersCount();

model.Teachers = await teacherService.GetAllTeachersWithStatusWaiting();

TempData["pages"] = count % 9 == 0 ? count / 9 : (count / 9) + 1; ;

TempData["curr"] = 0;

}

else if (filter == 1) // Students

{

model.Users = await studentService.GetAllStudents();

}

else if (filter == 2) // Teachers

{

model.Users = await teacherService.GetAllTeachers();

}

else //MonUsers

{

model.Users = await monUserService.GetAllMonUsers();

}

model.Teachers = await teacherService.GetAllTeachersWithStatusWaiting();

return View(model);

}

**3.2.3.** Ауторизация и оторизация на потребителите на приложението

* **Регистрация**

*Регистрацията може да бъде три вида. Всеки нерегистриран потребител може да се регистрира като ученик или учител. Регистрацията на потребител от МОН се извършва от администратора.*

* **Вход**

*За да влезеш в приложението трябва вече да си регистриран. Входа е с имейл и парола.*

* **Роли**

В приложението има 4 роли. Те са администратор, ученик, учител и потребител на МОН, като всяка от тях има различни права. Кодът, чрез който се създават ролите е:

public void Configure(EntityTypeBuilder<IdentityRole> builder)

{

builder.HasData(new List<IdentityRole>

{

new IdentityRole {

Id ="0f761db2-ab55-416c-83b9-70abded3d908" ,

Name = "Administrator",

NormalizedName = "ADMINISTRATOR"

},

new IdentityRole {

Id ="e66d730b-bcf1-41b5-b7e0-3e66056e61d9" ,

Name = "Student",

NormalizedName = "STUDENT"

},

new IdentityRole {

Id = "71281cf3-9730-4d7e-acbb-213edee8291c",

Name = "Teacher",

NormalizedName = "TEACHER"

},

new IdentityRole {

Id = "fe750b82-6fe9-472c-bdc5-61f5433d429e",

Name = "MonUser",

NormalizedName = "MONUSER"

},

});

}

**ЗАКЛЮЧЕНИЕ**

1. **Основен резултат**

Основните цели, описани по-горе са реализирани успешно. В това приложение учениците могат да се запознаят със свои връстници с подобни интереси и да си помагат. Разработеното WEB приложение, покрива напълно дейностите за библиотека, поддържа CRUD операции за книгите и осигурява на читателите възможност да разглеждат филтрирани резултати за книги и да заемат и връщат от наличните книги. Допълнително от заданието е реализиран форум и новини.

Идеите ми за бъдеща разработка са да разработя система за създаване на тестове за още по-добра подготовка за матурата.

1. **Реализация на приложението**

Приложението е публикувано в интернет, тествана е неговата функционалност от потребители и е готово за употреба.

Постепенното създаване на приложението може да бъде видяно в моят github акаунт – kresayuts05

**https://github.com/kresayuts05/PreparationForITExam**

1. **Заключение**

* В заключение, промените в учебната програма и изпитите за завършване на средното образование наложиха необходимостта от нов подход към подготовката за матурата по информатика. Разработването на уеб приложение "БлицКод-ДЗИ" представлява отговор на този предизвикателен контекст, като цели да предостави на учениците и учителите възможност за по-ефективно и систематично подготвяне. Включването на разнообразни ресурси и възможността за споделяне на материали ще допринесе за обогатяване на образователния процес и подобряване на успеваемостта на учениците. Вярвам, че този иновативен подход ще помогне за създаването на по-динамична и адаптивна образователна среда, която да отговаря на съвременните изисквания и нужди на учениците и учителите.
* Приложението е написано качествено, лесно е за използване и има светло бъдеще за реализация.

1. **Инсталиране**

За да се инсталира приложението, е необходимо да се изпълнят следните стъпки:

1. Отваряне на проекта във Visual Studio
2. Смяна на connection string в appsettings.json (PreparationForITExam)
3. Стартиране на приложението

***5******. Акаунти и пароли***

***1. Учител –*** [***teacher@gmail.com***](mailto:teacher@gmail.com) ***111111***

***2. Ученик –*** [***student@gmail.com***](mailto:student@gmail.com) ***111111***

***3. Мон –*** [***monuser@gmail.com***](mailto:monuser@gmail.com) ***111111***

***4. Администратор –*** [***admin@gmail.com***](mailto:admin@gmail.com) ***111111***

**6.** **Използвана литература**

**-** [**https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0**](https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0)

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

**-** [**https://learn.microsoft.com/en-us/ef/**](https://learn.microsoft.com/en-us/ef/)

**-** [**https://github.com/BG-IT-Edu/School-Programming/tree/main/Courses/Software-Sciences**](https://github.com/BG-IT-Edu/School-Programming/tree/main/Courses/Software-Sciences)

**-** [**https://ej2.syncfusion.com/aspnetmvc/documentation/introduction**](https://ej2.syncfusion.com/aspnetmvc/documentation/introduction)

**-** [**https://cloudinary.com/documentation/image\_upload\_api\_reference**](https://cloudinary.com/documentation/image_upload_api_reference)

**7.** **Електронни ресурси:**

* <https://www.w3schools.com/>
* <https://stackoverflow.com/>

*ПРИЛОЖЕНИЯ:*

A computer screen shot of a computer

Description automatically generated