Санкт-Петербургское государственное бюджетное профессиональное образовательное учреждение

«Радиотехнический колледж»

К защите допущен

заместитель директора по УПР

\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Бессчетнова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ г.

мп

**ДИПЛОМНЫЙ ПРОЕКТ**

Тема: «Разработка системы управления учебным временем студента»

**Специальность:** 09.02.07 «Информационные системы и программирование»

**Выполнил(а):** обучающийся группы  301  Анастасия Вадимовна Писукова

(номер группы)

**Руководитель:** преподаватель Виталий Сергеевич Кондратьев\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Санкт-Петербург 2023 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc137580442)

[1. ПРЕДМЕТНАЯ ОБЛАСТЬ И АНАЛОГИ 5](#_Toc137580443)

[1.1. Системный анализ предметной области. 5](#_Toc137580444)

[1.2. Обзор аналогов. 6](#_Toc137580445)

[2. ПРОЕКТИРОВАНИЕ ИС 10](#_Toc137580446)

[2.1. Описание этапов проектирования системы. 10](#_Toc137580447)

[2.2. Проектирование структуры ИС. 11](#_Toc137580448)

[2.3. Проектирование базы данных. 16](#_Toc137580449)

[2.4. Проектирование дизайна сайта. 24](#_Toc137580450)

[2.5. Даталогическое проектирование базы данных. 27](#_Toc137580451)

[2.6. Выбор средств реализации проекта. 28](#_Toc137580452)

[3. ПОЭТАПНОЕ ОПИСАНИЕ РЕАЛИЗАЦИИ КОМПОНЕНТОВ ИС 31](#_Toc137580453)

[3.1. Подготовка. 31](#_Toc137580454)

[3.2. Создание ролей с помощью RBAC. 32](#_Toc137580455)

[3.3. Страницы авторизации и регистрации. 34](#_Toc137580456)

[3.4. Главная страница и личный кабинет. 36](#_Toc137580457)

[3.5. Управление задачами. 37](#_Toc137580458)

[3.6. Управление ответами на задачи. 41](#_Toc137580459)

[3.7. Управление группами. 43](#_Toc137580460)

[3.8. Управление академическими предметами. 43](#_Toc137580461)

[3.9. Панель администрирования. 44](#_Toc137580462)

[3.10. Руководство пользователя. 45](#_Toc137580463)

[3.11. Руководство программиста 52](#_Toc137580464)

[4. ЭКОНОМИЧЕСКАЯ ЧАСТЬ 53](#_Toc137580465)

[4.1. Определение затрат: 53](#_Toc137580466)

[4.2. Расчет себестоимости программного продукта 53](#_Toc137580467)

[ЗАКЛЮЧЕНИЕ 56](#_Toc137580468)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 58](#_Toc137580469)

[ПРИЛОЖЕНИЕ А (обязательное) Программные коды 60](#_Toc137580470)

## ВВЕДЕНИЕ

Актуальность.

Тема дипломного проекта: разработка системы управления учебным временем студента. Актуальность выбранной темы для дипломной работы очевидна – в современном мире личное время является ключевым ресурсом для любого человека, не говоря уже об обучающихся, осваивающих образовательную программу среднего и высшего профессионального образования. Умение продуктивно организовывать свой досуг представляет для студента высокую ценность.

Ни для кого не секрет, что в погоне за знаниями студенты осуществляют работу в режиме многозадачности. Важность решения обучающимся множества задач и проблем, связанных с постоянным самосовершенствованием и получением профессиональных навыков в условиях катастрофической нехватки времени, может привести к его переутомлению, поставить под угрозу не только здоровье, но и непосредственно сам процесс обучения.

На сегодняшний день в различных образовательных учреждениях создаются интернет-ресурсы, направленные на регулирование рабочего и учебного времени, поскольку использование и владение технологиями тайм-менеджмента может оказать неоценимую помощь в любой деятельности. Современные технологии организации времени позволяют использовать невосполнимые часы и минуты жизни студента в соответствии с его целями и ценностями. Грамотное управление временным ресурсом также позволит каждому учащемуся рационально, последовательно и спокойно выполнять свои учебные обязанности.

Объект исследования.

Объектом исследования является специфика процесса проектирования и создания информационной системы управления учебным временем студента.

Предмет исследования.

Предметом исследования является проектирование и создание информационной системы.

**Цель работы**

Целью дипломного проекта является разработка системы управления учебным временем, обеспечивающей удобство организации процесса выполнения запланированной работы студента, а также безопасность и интуитивную понятность интерфейса пользователю.

**Задачи**

На основе поставленной цели были определены семь задач работы:

1. Анализ предметной области и конкурентов,
2. Определение требований к информационной системе,
3. Выбор средств реализации проекта,
4. Проектирование структуры системы и базы данных,
5. Разработка дизайна системы,
6. Создание баз данных,
7. Реализация компонентов системы.

## 1. ПРЕДМЕТНАЯ ОБЛАСТЬ И АНАЛОГИ

1.1. Системный анализ предметной области.

Область деятельности – разрабатываемая мной система предназначена для организации учебного времени студента. Эта система должна помочь в планировании задач для обучающегося, что в свою очередь является необходимым условием для повышения уровня обучения в образовательной организации.

На сегодняшний день, учебный процесс имеет ряд существенных недостатков:

* сложность структурирования информации (из-за большого объема учебных материалов);
* большие затраты времени при составлении плана по выполнению учебных задач;
* трудности с получением информации при необходимости работы с данной информацией.

Для решения данных проблем необходимо автоматизировать процесс организации досуга обучающегося с помощью информационной системы.

Информационная система должна обеспечивать студентам:

* возможность удобного структурированного хранения информации в объеме существующих документов;
* возможность удобного и быстрого ввода и занесения необходимой информации;
* возможность быстрого вывода и архивирования информации; возможность сетевого доступа к информационной системе всех заинтересованных в получении информации пользователей;
* ограничение доступа к имеющейся в информационной системе конфиденциальной информации через разграничение прав доступа пользователей (требование законодательства о защите персональных данных);
* отдельное хранение документов с возможностью быстрого обращения к ним и последующего открытия;
* быструю работу и взаимодействие студента и куратора группы;
* использование различных техник тайм-менеджмента для более продуктивной работы.

1.2. Обзор аналогов.

Системы для управления рабочим или учебным временем на сегодняшний день стали популярными инструментами для совместной работы, используемыми организациями по всему миру. Тем не менее люди часто осознают, что у предложенных программ есть ряд ограничений и они не обладает подходящим набором возможностей для их типичного клиента, а так как на рынке сейчас много схожих решений, то пользователи начинают искать альтернативы.

Для примера можно выбрать несколько популярных сайтов, подходящих по тематике и произвести некоторый анализ конкурентного рынка, по результатам которого можно будет судить о том, какие функциональные опции необходимо разработать для получения превосходства над сравниваемыми схожими проектами.

Чтобы упростить поставленную задачу, для анализа выбираются три ближайшие конкурента:

1. Программа для планирования и управления задачами Todoist.

Todoist— одна из самых известных программ данного вида. Она предоставляет возможности для управления всеми видами задач. Таск-менеджер помогает расставлять приоритеты, а также легко создавать задачи, подзадачи и разделы. В программе можно оставлять комментарии, следить за прогрессом и расставлять приоритеты.

Отличительной чертой этого таск-менеджера является игровой способ работы. Использование геймификации мотивирует пользователей оставаться продуктивными, зарабатывая различные награды, очки и переходя на новые уровни. На рисунке 1 представлен внешний вид приложения.

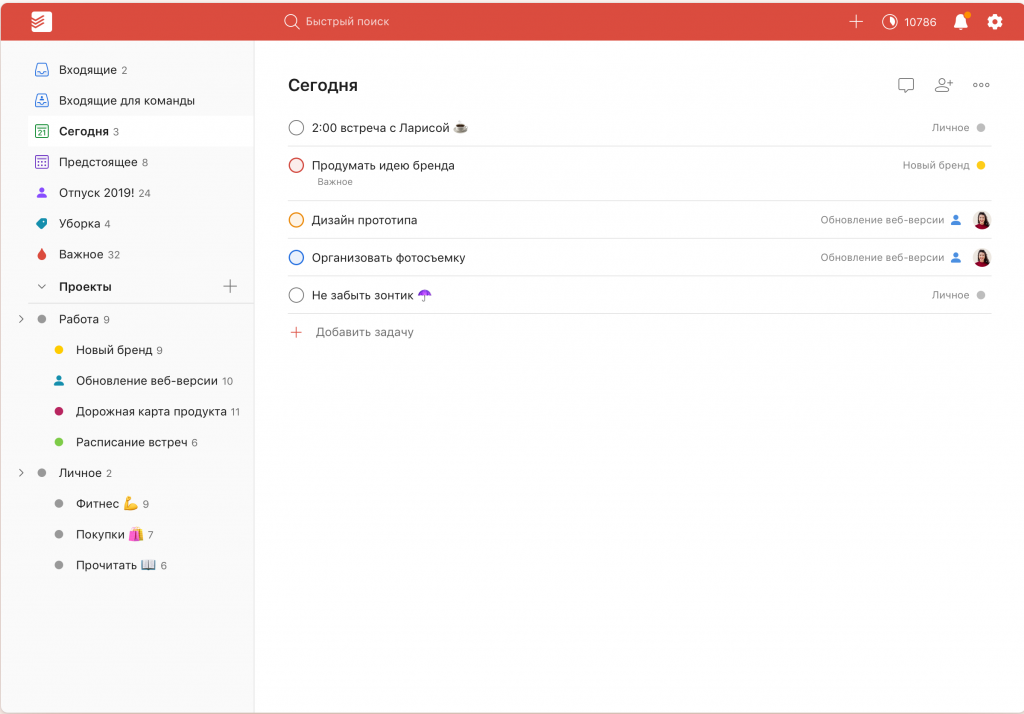


Рисунок 1 – Программа Todoist

1. Сервис для управления проектами Shtab.

Shtab – готовое решение для бизнеса по управлению проектами. Оно включает в себя функцию управления задачами (таск-трекер), аналитику по активности, учёт зарплаты, отчёты по проектам, функцию отслеживания времени (тайм-трекер).

В интерфейсе приложения нетрудно ориентироваться среди множества задач и проектов. Все участники команды видят обновления, понимают в какой стадии проект и быстрее реагируют на изменения. Представление одной из страниц сервиса изображено на рисунке 2.

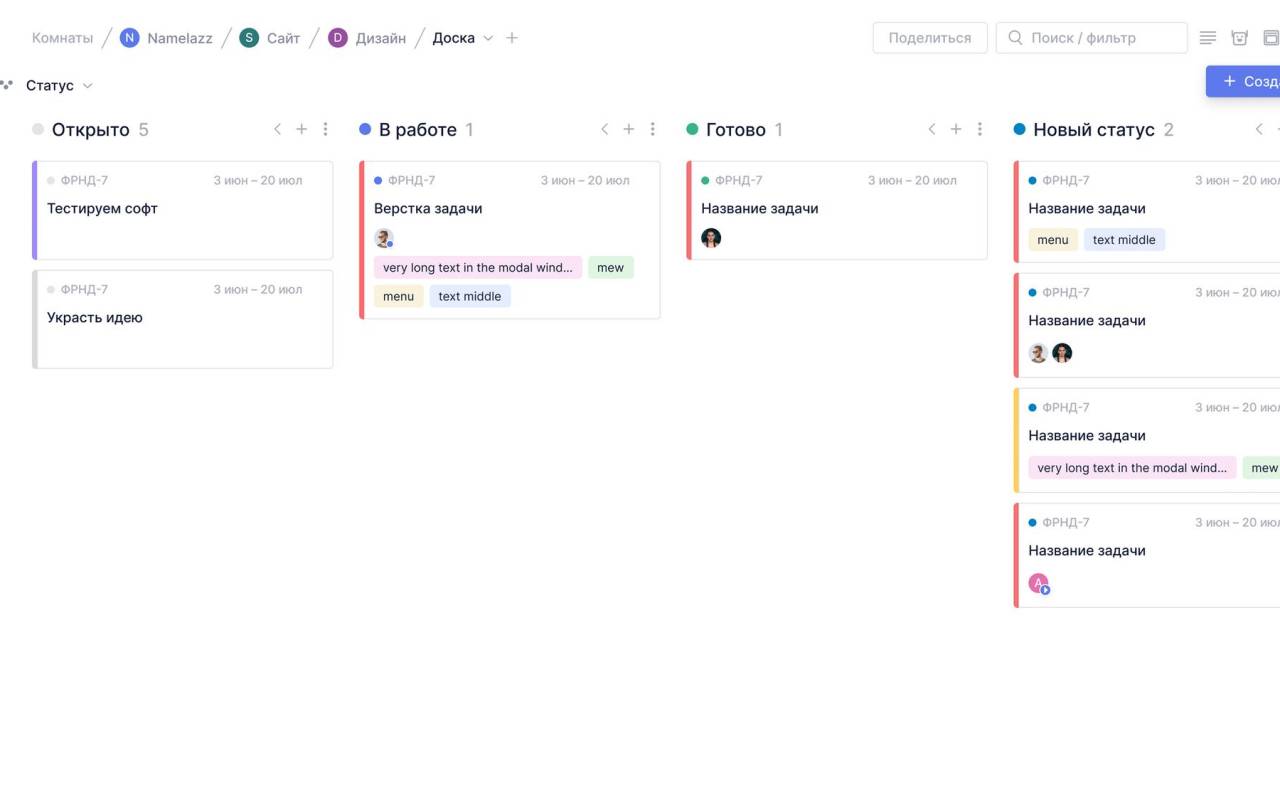


Рисунок 2 – Сервис Shtab

1. Сервис управления делами ЛидерТаск.

Цифровой продукт российских разработчиков ЛидерТаск отвечает всем требованиям как для персонального использования, так и для работы в команде. Основной функционал сервиса:

* составление списков дел на каждый день, разделение их на подзадачи для более успешного выполнения сложных задач;
* создание повторяющихся задач;
* встроенный в каждое приложение календарь;
* напоминания о предстоящих встречах и задачах;
* чат для обсуждения деталей выполнения задачи.

Платформа оснащена удобным пользователю интерфейсом, пример которого можно увидеть на рисунке 3.

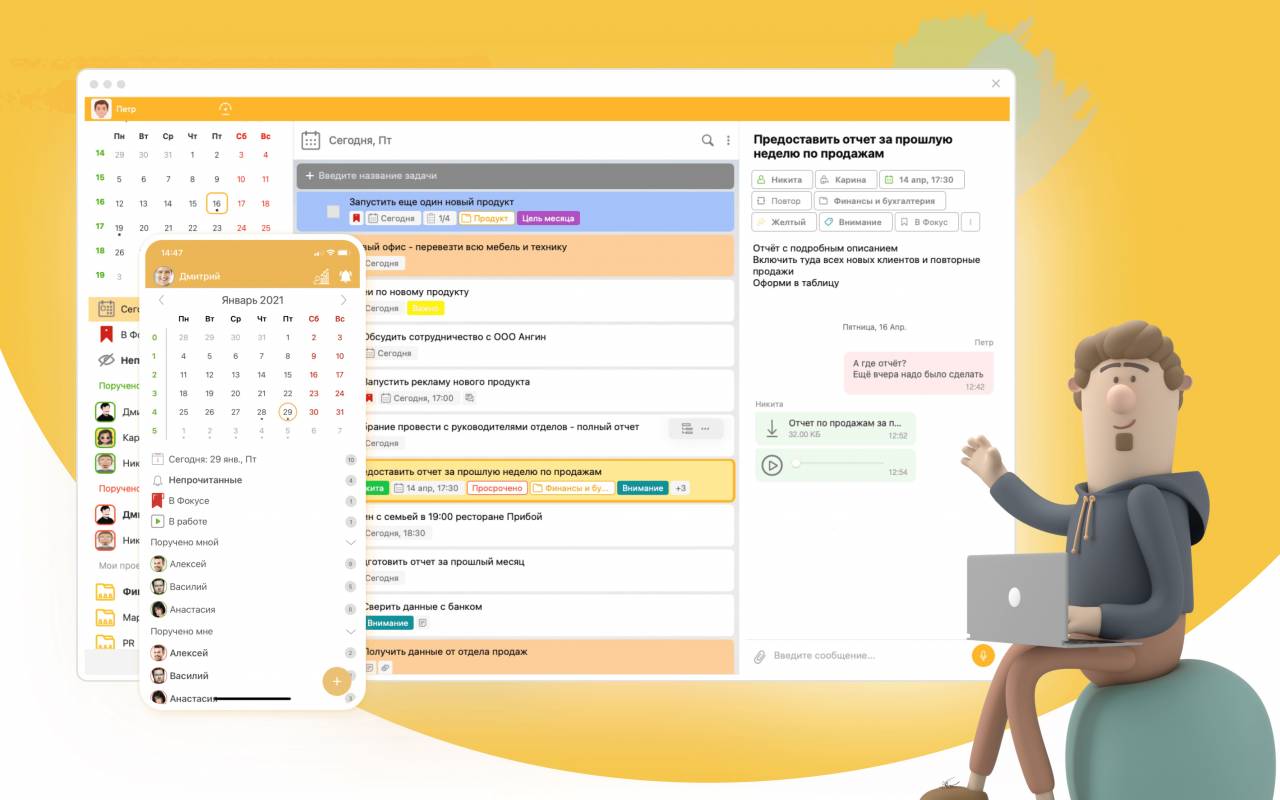


Рисунок 3 – Сервис ЛидерТаск

Изучив информационные системы конкурентов, можно выделить несколько критериев, устанавливающих преимущества или же недостатки перечисленных платформ. С помощью указанных критериев составлена таблица 1, в которой столбцами обозначены сайты конкурентов, а строками -критерии.

Таблица 1 – Обзор аналогов.

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Todoist | Shtab | ЛидерТаск |
| Стоимость | Бесплатно, несколько премиум версий от 316 руб./мес. | Бесплатно, несколько премиум версий от 130 руб./мес. | Бесплатно по подписке, премиум-версия 249 руб./мес. |
| Простота в использовании, интуитивный интерфейс | Да | Да | Да |
| Поддержка русского языка | Да | Да | Да |
| Автоматические уведомления | Да | Да | Да |
| Фильтрация и сортировка задач | Нет | Да | Нет |
| Статистика выполнения задач | Нет | Да | Нет |

Исходя из проведенного исследования следует, что при добавлении некоторых анализируемых функциональных опций, например возможности просматривать статистику выполнения пользователем своих задач, может получиться продукт, совмещающий в себе лишь сильные стороны схожих проектов, что даст разрабатываемому веб-сервису огромное преимущество среди конкурентов в сфере систем управления и тайм-менеджмента. Учитывая целевую аудиторию проектируемой системы, появляется возможность для осуществления уникального проекта, направленного на корпоративное использование.

## 2. ПРОЕКТИРОВАНИЕ ИС

2.1. Описание этапов проектирования системы.

В системе должна быть реализована регистрация и авторизация пользователя. Эти две подсистемы служат для обеспечения парольной защиты и разграничения прав доступа в систему. С помощью регистрации устанавливаются некоторые роли пользователей, которые в дальнейшем будут разграничивать доступ к функционалу. Распределение пользователей по ролям показано на рисунке 4.

Роли

Администратор

Студент

Куратор

Заместитель директора

Неавторизованный пользователь

Пользователь системы

Рисунок 4 – Роли в системе

В системе должны быть представлены следующие основные разделы, распределённые по ролям пользователей:

Неавторизованный пользователь:

* авторизация.

Студент:

* регистрация,
* управление задачами (просмотр, создание, редактирование, выполнение, удаление),
* управление ответами на задачи для группы (просмотр, отправка, редактирование, отмена),
* управление личным кабинетом,
* выход из системы.

Куратор:

* регистрация,
* управление задачами (просмотр, создание, редактирование, выполнение, удаление),
* управление ответами на задачи для группы (просмотр, проверка),
* просмотр курирующих групп,
* управление студентами (просмотр, добавление, удаление),
* управление личным кабинетом,
* выход из системы.

Заместитель директора:

* управление задачами (просмотр, создание, редактирование, выполнение, удаление),
* управление ответами на задачи для группы (просмотр, проверка),
* управление группами (просмотр, создание, редактирование, удаление),
* управление предметами (просмотр, создание, редактирование, удаление),
* управление личным кабинетом,
* выход из системы.

Администратор:

* управление группами (просмотр, создание, редактирование, удаление),
* управление предметами (просмотр, создание, редактирование, удаление),
* управление пользователями (просмотр, создание, редактирование, удаление, присваивание пользователям ролей),
* выход из системы.

2.2. Проектирование структуры ИС.

Проектируемая система должна включать в себя управление задачами. При создании новой задачи к ней допускается добавление файлов, учебного предмета, установка срока выполнения, а также перевод задачи в статус «важная». Доступен функционал редактирования, выполнения и удаления. Каждый пользователь может сортировать и фильтровать задачи по статусу, приоритету, академическому предмету. В дополнение к этому, на странице будет представлена фильтрация данных по дате дедлайна с помощью календаря.

В современной педагогике появился такой термин, как рефлексия - размышление о своих чувствах, анализ собственных действий и их причин. Исследование Гарвардской школы бизнеса показало, что рефлексирующие сотрудники кол‑центров работают на 23% лучше тех, кто не пользуется данным методом самоанализа. Первые быстрее понимали, что от них требуется, и увереннее принимали решения. Способность к рефлексии является очень важным навыком в наши дни, поэтому было решено включить ее в проект как оценочную форму выполнения задач. Этот модуль будет представлен всплывающим модальным окном после выполнения задачи. В нём пользователю нужно будет выбрать эмоцию, которую он испытывает при совершении действия. Такая работа над ошибками поможет стать гораздо эффективнее.

Чтобы облегчить обмен информацией между студенческими группами и их руководителями, а также заместителем директора, было решено ввести в проект дополнительный инструментарий, обеспечивающий связь этих ролей. Таким образом, куратор и заместитель директора смогут создавать задачи и назначать их студентам определенной группы. Каждый студент этой группы увидит указанную задачу и сможет прикрепить к ней ответ, содержащий в себе текст и необходимый документ. Создатель задачи, в свою очередь, имеет возможность просмотреть эти ответы и проверить их, кроме того, ему будет доступна сортировка и фильтрация ответов по статусу.

Для лучшего понимания структуры системы спроектирована use-case диаграмма, изображенная на рисунке 5.

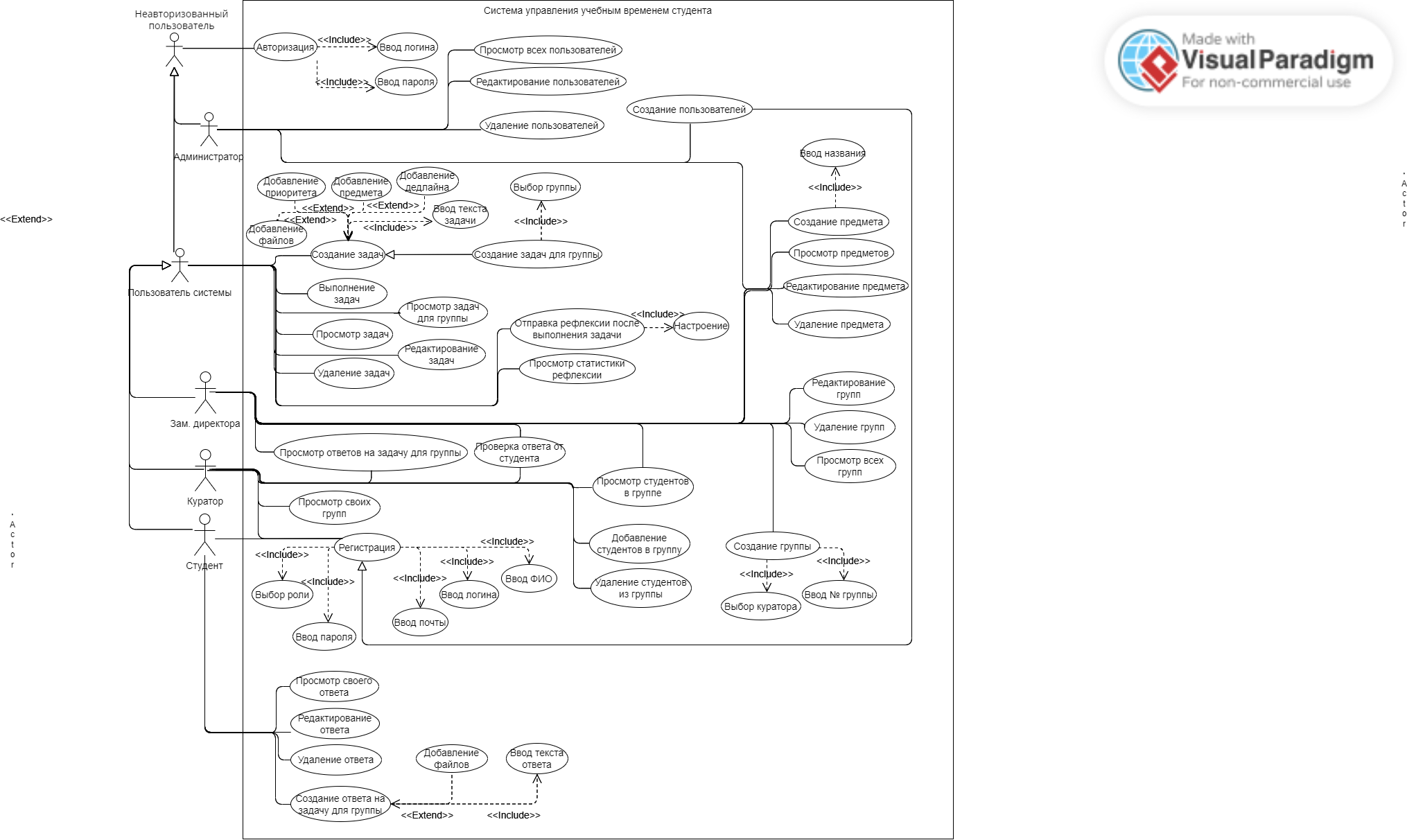


Рисунок 5 – Use-case диаграмма

Роль «Куратор» включает в себя модуль работы с курирующими группами, а также возможности управления студентами своих групп. Модуль работы с группой позволяет куратору осуществить просмотр групп, к которым он непосредственно принадлежит.

Подсистема заместителя директора предназначена для управления группами с целью донесения до студентов важной и актуальной информации. Она включает в себя модуль работы с группами, работу с предметами. Модуль работы с группами позволяет осуществить создание, редактирование и просмотр всех существующих групп. Модуль работы с предметами позволяет добавлять, удалять и открывать для просмотра академические предметы. Созданные в системе предметы будут в дальнейшем служить для обозначения принадлежности предмету какой-либо задачи.

Подсистема администратора состоит из трех модулей: работа с предметами, работа с группами и работа с пользователями. Модуль работы с предметами позволяет администратору добавлять, удалять и просматривать предметы. Модуль работы с пользователями позволяет создавать, удалять и редактировать пользователей. При создании нового пользователя или редактировании существующего, администратор вправе назначить ему любую роль из списка ролей. Модуль работы с группами позволяет создавать, удалять или редактировать группы.

Чтобы наглядно продемонстрировать интерфейс приложения, были разработаны несколько вариантов будущей системы.

Форма «Авторизация» содержит текстовые поля для ввода логина и пароля и кнопку «Войти». Внешний вид формы представлен на рисунке 6.

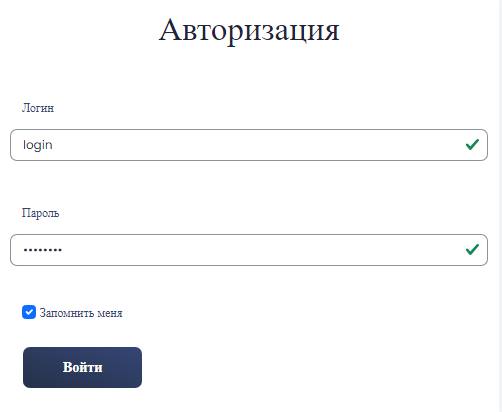


Рисунок 6 – Форма «Авторизация»

Форма «Регистрация» обеспечивает возможность регистрации нового пользователя системы. В форме присутствует чек-бокс «Я студент», позволяющий разграничить права куратора и студента (рисунок 7).

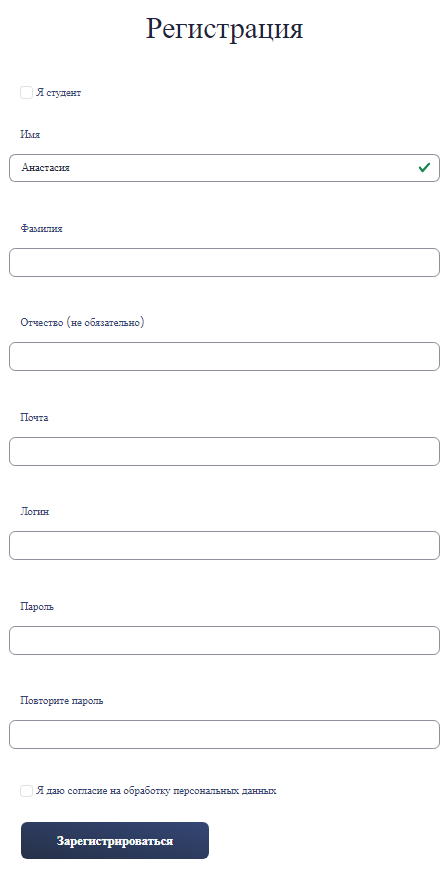


Рисунок 7 – Форма «Регистрация»

Группа элементов «Работа с пользователями» в правах администратора содержит кнопки создания, просмотра, редактирования и удаления. Страница со списком пользователей показана на рисунке 8.

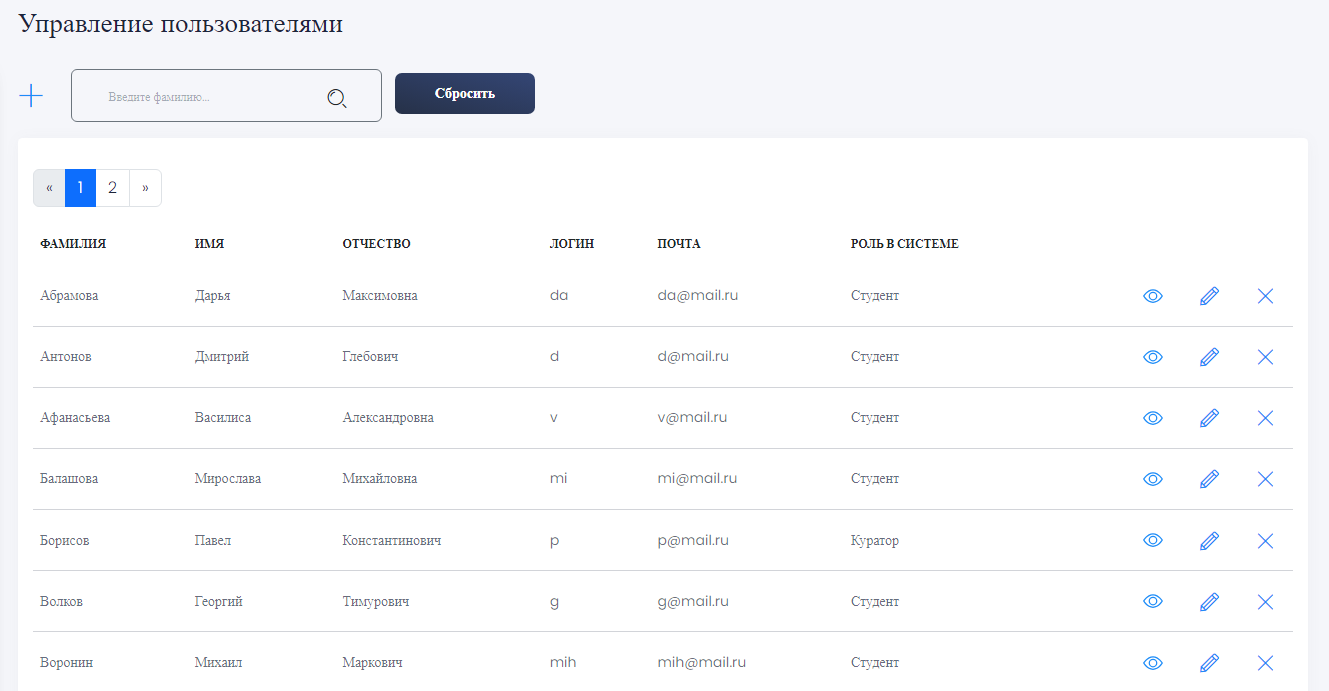


Рисунок 8 – Раздел «Управление пользователями»

Группы элементов «Работа с группой» и «Работа с академическими предметами» имеют сходный интерфейс: таблицу, в которую выводится информация из выбранной таблицы соответствующей базы данных, и кнопки управления информацией. Такими же таблицами будет оборудована каждая из перечисленных мной выше подсистем.

2.3. Проектирование базы данных.

Выполним концептуальное проектирование базы данных системы, необходимое для структурирования элементов при ее разработке. На этапе проектирования системы было распределено несколько ролей и выделены модули системы. Вследствие этого, база данных будет включать в себя упомянутые выше сущности, их связи и атрибуты.

Для начала в базе данных была создана таблица пользователей – user. В таблице 2 указаны поля таблицы user.

Таблица 2 – Таблица user.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **По умолчанию** | **Ключевое поле** |
| id | int(10) | UNSIGNED |  | auto\_increment |
| name | varchar(255) |  |  |  |
| surname | varchar(255) |  |  |  |
| patronymic | varchar(255) |  | NULL |  |
| login | varchar(255) |  |  |  |
| email | varchar(255) |  |  |  |
| password | varchar(255) |  |  |  |
| auth\_key | varchar(255) |  |  |  |

В разрабатываемой системе используется RBAC (контроль доступа на основе ролей), благодаря этому механизму автоматически создаются таблицы в базе данных, которые отвечают за разграничение ролей.

Инструмент RBAC использует пять таблиц для хранения данных:

1. auth\_rule: таблица для хранения правил, данные представлены в таблице 3.
2. auth\_item: таблица для хранения авторизационных элементов, данные представлены в таблице 4.
3. auth\_item\_child: таблица для хранения иерархии элементов, данные представлены в таблице 5.
4. auth\_assignment: таблица для хранения назначений элементов авторизации, данные представлены в таблице 6.
5. migration: таблица для хранения миграций, данные представлены в таблице 7.

Таблица 3 – Таблица auth\_rule.

|  |  |  |
| --- | --- | --- |
| **Имя столбца** | **Тип** | **По умолчанию** |
| name | varchar(64) |  |
| data | blob | NULL |
| created\_at | int(11) | NULL |
| updated\_at | int(11) | NULL |

Таблица 4 – Таблица auth\_item.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **По умолчанию** | **Связи** |
| name | varchar(64) |  |  |
| title | varchar(255) | NULL |  |
| type | smallint(6) |  |  |
| description | text | NULL |  |
| rule\_name | varchar(64) | NULL | Таблица auth\_rule, атрибут name |
| data | blob | NULL |  |
| created\_at | int(11) | NULL |  |
| updated\_at | int(11) | NULL |  |

Таблица 5 – Таблица auth\_item\_child.

|  |  |  |
| --- | --- | --- |
| **Имя столбца** | **Тип** | **Связи** |
| parent | varchar(64) | Таблица auth\_item, атрибут name |
| child | varchar(64) | Таблица auth\_item, атрибут name |

Таблица 6 – Таблица auth\_assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **По умолчанию** | **Связи** |
| item\_name | varchar(64) |  | Таблица auth\_item, атрибут name |
| user\_id | varchar(64) |  |  |
| created\_at | int(11) | NULL |  |

Таблица 7 – Таблица migration.

|  |  |  |
| --- | --- | --- |
| **Имя столбца** | **Тип** | **По умолчанию** |
| version | varchar(180) |  |
| apply\_time | int(11) | NULL |

Была создана таблица group, в которой хранятся данные о группах, а также вспомогательная таблица stud\_of\_group, которая помогает определить принадлежность студентов к группам. В таблице 8 и 9 указаны поля упомянутых таблиц.

Таблица 8 – Таблица group.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** | **Связи** |
| id | int(10) | UNSIGNED | auto\_increment |  |
| title | varchar(255) |  |  |  |
| manager\_id | int(10) | UNSIGNED |  | Таблица user, атрибут id |

Таблица 9 – Таблица stud\_of\_group.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** | **Связи** |
| id | int(10) | UNSIGNED | auto\_increment |  |
| group\_id | int(10) | UNSIGNED |  | Таблица group, атрибут id |
| student\_id | int(10) | UNSIGNED |  | Таблица user, атрибут id |

Таблица priority предназначена для определения приоритетов задач. В таблице 10 указаны поля таблицы priority.

Таблица 10 – Таблица priority.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** |
| id | int(10) | UNSIGNED | auto\_increment |
| title | varchar(255) |  |  |

Для осуществления метода рефлексии создана таблица reflection, имеющая связь к таблице mood, которая содержит список настроений. В таблице 11 и 12 указаны поля созданных таблиц.

Таблица 11 – Таблица reflection.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **По умолчанию** | **Ключевое поле** | **Связи** |
| id | int(10) | UNSIGNED |  | auto\_increment |  |
| date | timestamp |  | current\_timestamp() | on update current\_timestamp() |  |
| user\_id | int(10) | UNSIGNED |  |  | Таблица user, атрибут id |
| task\_id | int(10) | UNSIGNED |  |  | Таблица task, атрибут id |
| mood\_id | int(10) | UNSIGNED |  |  | Таблица mood, атрибут id |

Таблица 12 – Таблица mood.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** |
| id | int(10) | UNSIGNED | auto\_increment |
| title | varchar(255) |  |  |
| image | varchar(255) |  |  |

Таблица status предназначена для определения статусов задач и ответов. В таблице 13 указаны поля таблицы status.

Таблица 13 – Таблица status.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** |
| id | int(10) | UNSIGNED | auto\_increment |
| title | varchar(255) |  |  |

Таблица subject содержит данные об академических предметах. В таблице 14 указаны поля таблицы subject.

Таблица 14 – Таблица subject.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** |
| id | int(10) | UNSIGNED | auto\_increment |
| title | varchar(255) |  |  |

Таблица task содержит данные о задачах, созданных пользователями, а также имеет связи с таблицами user, group, status, priority, subject. Поле checked предназначено для корректной работы функционала выполнения задачи. В таблице 15 указаны поля таблицы task.

Таблица 15 – Таблица task.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **По умолчанию** | **Ключевое поле** | **Связи** |
| id | int(10) | UNSIGNED |  | auto\_increment |  |
| title | varchar(255) |  |  |  |  |
| date | date |  |  |  |  |
| deadline\_date | date |  | NULL |  |  |
| deadline\_time | time |  | NULL |  |  |
| user\_id | int(10) | UNSIGNED |  |  | Таблица user, атрибут id |
| group\_id | int(10) | UNSIGNED | NULL |  | Таблица group, атрибут id |
| status\_id | int(10) | UNSIGNED |  |  | Таблица status, атрибут id |
| priority\_id | int(10) | UNSIGNED |  |  | Таблица priority, атрибут id |
| subject\_id | int(10) | UNSIGNED | NULL |  | Таблица subject, атрибут id |
| checked | tinyint(1) |  | 0 |  |  |

В таблице response содержатся данные об ответах студентов на задачи, предназначенные для групп, в которых они состоят. В таблице 16 указаны поля таблицы response.

Таблица 16 – Таблица response.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **По умолчанию** | **Ключевое поле** |
| id | int(10) | UNSIGNED |  | auto\_increment |
| text | text |  | NULL |  |
| date | datetime |  | current\_timestamp() | on update current\_timestamp() |
| task\_id | int(10) | UNSIGNED |  |  |
| student\_id | int(10) | UNSIGNED |  |  |
| status\_id | int(10) | UNSIGNED |  |  |

Была создана таблица file, которая хранит в себе загружаемые пользователями файлы для последующего скачивания. В таблице 17 указаны поля таблицы file.

Таблица 17 – Таблица file.

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** |
| id | int(10) | UNSIGNED | auto\_increment |
| name | varchar(255) |  |  |

Таблица task\_files связывает созданные пользователем задачи и прикрепленные к ним файлы. В таблице 18 указаны поля таблицы task\_files. Таким же образом сформирована таблица response\_files, поля которой указаны в таблице 19.

Таблица 18 – Таблица task\_files.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** | **Связи** |
| id | int(10) | UNSIGNED | auto\_increment |  |
| task\_id | int(10) | UNSIGNED |  | Таблица task, атрибут id |
| file\_id | int(10) | UNSIGNED |  | Таблица file, атрибут id |

Таблица 19 – Таблица response\_files.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Тип** | **Атрибуты** | **Ключевое поле** | **Связи** |
| id | int(10) | UNSIGNED | auto\_increment |  |
| response\_id | int(10) | UNSIGNED |  | Таблица response, атрибут id |
| file\_id | int(10) | UNSIGNED |  | Таблица file, атрибут id |

Для наглядности, в сервисе [Draw.io](https://app.diagrams.net/) мной была создана диаграмма «Сущность-связь» спроектированной базы данных, представленная на рисунке 9.

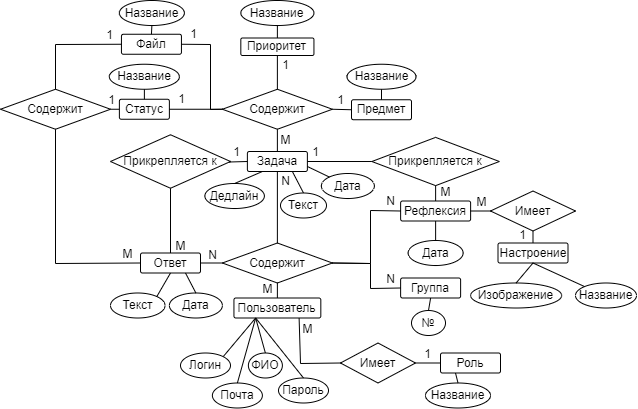


Рисунок 9 - Диаграмма «Сущность-связь»

Графически сущности на диаграммах «сущность-связь» представляются в виде прямоугольников, изображение атрибутов сущности представляются в виде овальных блоков, а для графического представления связи используется ромб. Показатель кардинальности связи (характеристика однозначности) обозначает степень взаимосвязи сущностей и описывает количество возможных связей для каждой из сущностей-участниц:

* один-к-одному (1:1),
* один-ко-многим (1:N),
* многие-ко-многим (N:M).

Была создана полная атрибутивная модель, полностью удовлетворяющая процессу нормализации данных. В результате проведения нормализации была создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам – формализованным требованиям к организации данных.

2.4. Проектирование дизайна сайта.

Страницы информационной системы подразделены на логические и тематические блоки в зависимости от их наполнения. Для быстрой вёрстки HTML-кода веб-приложения был выбран HTML, CSS и JS фреймворк Bootstrap. Благодаря наличию у него большого количества готовых шаблонов, можно создавать сайт с красивым, приятным, профессиональным дизайном за достаточно короткое время. Поэтому, для стилизации системы был подобран бесплатный шаблон «Black Dashboard» с открытым исходным кодом в сервисе BootstrapTema.

В процессе проектирования был выбран творческий подход к внешнему виду веб-приложения, поэтому был разработан персонаж - маскот, изображение которого поместится на множество страниц системы. Маскотов используют многие компании в своих интернет-порталах. Он является уникальным героем-талисманом организации, который представляет товары и услуги компаний и является эффективным инструментом маркетинга. В связи с этим, внедрение персонажа в разрабатываемую систему задумано для того, чтобы подчеркнуть её индивидуальность и выделить среди конкурентов. Также, маскот прекрасно подходит для знакомства с функционалом системы и последующей работы в ней в игровой форме.

Фирменный персонаж — это продуманный образ героя. Его наделяют определённым характером и разрабатывают для него уникальные детали-атрибуты, которые помогают с первого взгляда понять этот характер и сделать образ запоминающимся. Разработка маскота для бренда начинается с определения его характера, фирменных характеристик и задач в коммуникации с потенциальным потребителем. В системе управления учебным временем студента герой станет другом и наставником, будет учить, развлекать, подбадривать — в общем, поддерживать дружеское общение с аудиторией.

Для того, чтобы определиться с характеристиками маскота, необходимо выявить желания аудитории, провести анализ её ценностей и ожиданий от продукта. На основе исследования выявлено, что целевая аудитория системы – студенты средних и высших профессиональных учебных заведений, а также их сотрудники. Основная задача готового программного продукта состоит в том, чтобы сделать процесс обучения более эффективным и простым. В связи с этим, персонаж должен быть человекоподобным, олицетворять успешного и продуктивного обучающегося и транслировать такие важные характеристики, как доброта и дружелюбие.

Опираясь на проведенный анализ, был произведен уникальный герой по имени Мип – пришелец с планеты Мум (что расшифровывается как мастера управления и менеджмента). Мип поделится с пользователями своими знаниями в области тайм-менеджмента, а также поможет будущим специалистам сделать первые шаги в сторону улучшения качества их работы.

Далее, были нарисованы карандашные наброски маскота, и с помощью растрового графического редактора для рисования с открытым исходным кодом Krita были созданы иллюстрации персонажа по нарисованным ранее эскизам. Готовая иллюстрация продемонстрирована на рисунке 10.



Рисунок 10 – Фирменный персонаж системы

Так как в системе реализован модуль рефлексии, маскот необходимо было разработать еще и для того, чтобы иметь наглядное представление выбираемых пользователем эмоций. Исходя из этого, в проект добавились еще пять иллюстраций фирменного героя, показывающих эмоции радости, усталости, грусти, злости и беспокойства. Все эти чувства проиллюстрированы на рисунке 11.



Рисунок 11 – Эмоции фирменного персонажа

Нарисованные иллюстрации эмоций будут расположены в модальном окне, которое отображается при выполнении задачи. С помощью анимации и Css стилей, выбранный блок будет визуально выделяться, как показано на рисунке 12.



Рисунок 12 – Эмоции фирменного персонажа

После небольшого видоизменения цветовой палитры скачанного ранее шаблона, внедрения в него фирменного персонажа, а также исключения ненужных элементов и блоков, получился готовый дизайн, одна из страниц которого представлена на рисунке 13.

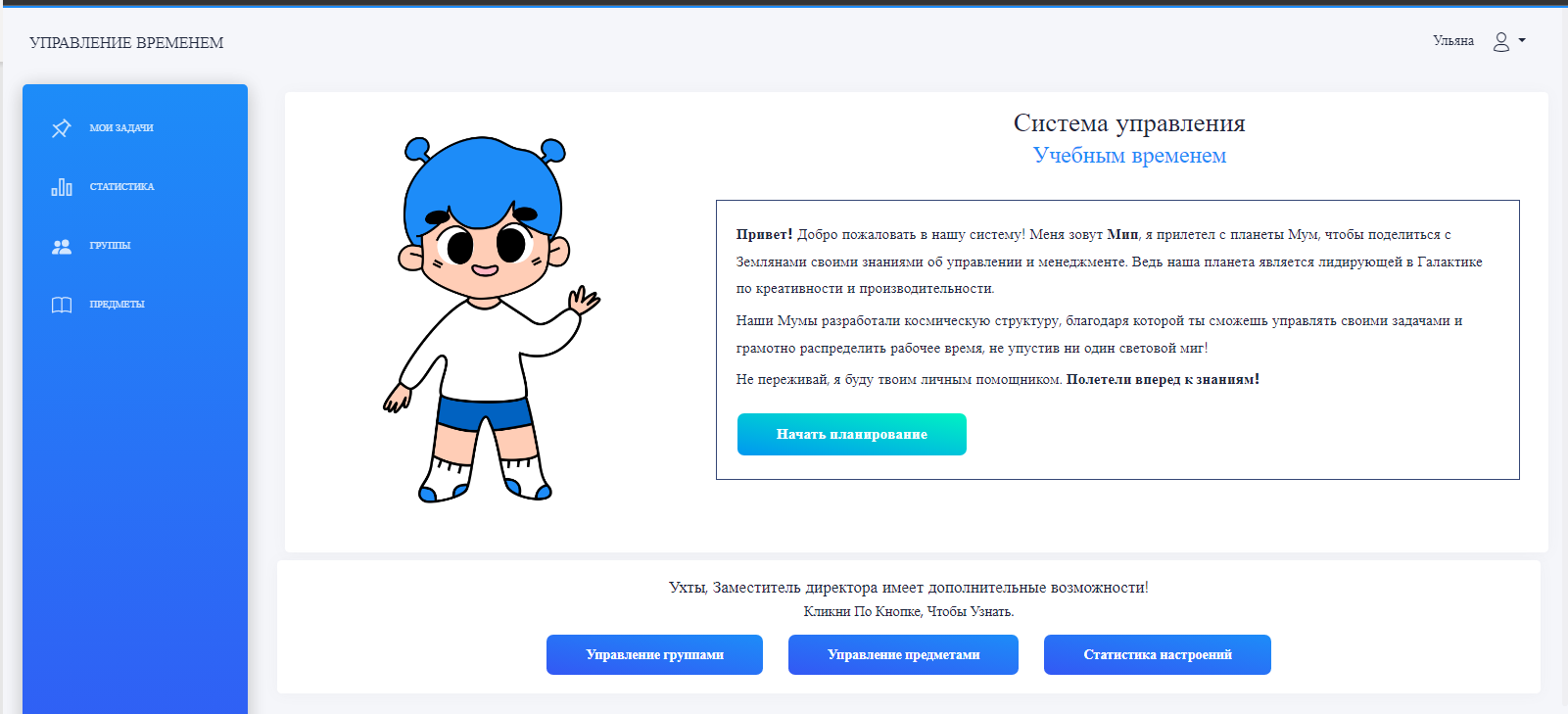


Рисунок 13 – Дизайн информационной системы

2.5. Даталогическое проектирование базы данных.

Разработка базы данных является непростой и трудоемкой задачей, так как именно она считается одним из самых важных элементов для информационной системы. При разработке учитывался проведенный ранее предпроектный анализ.

Разработанная база данных изображена на рисунке 14.

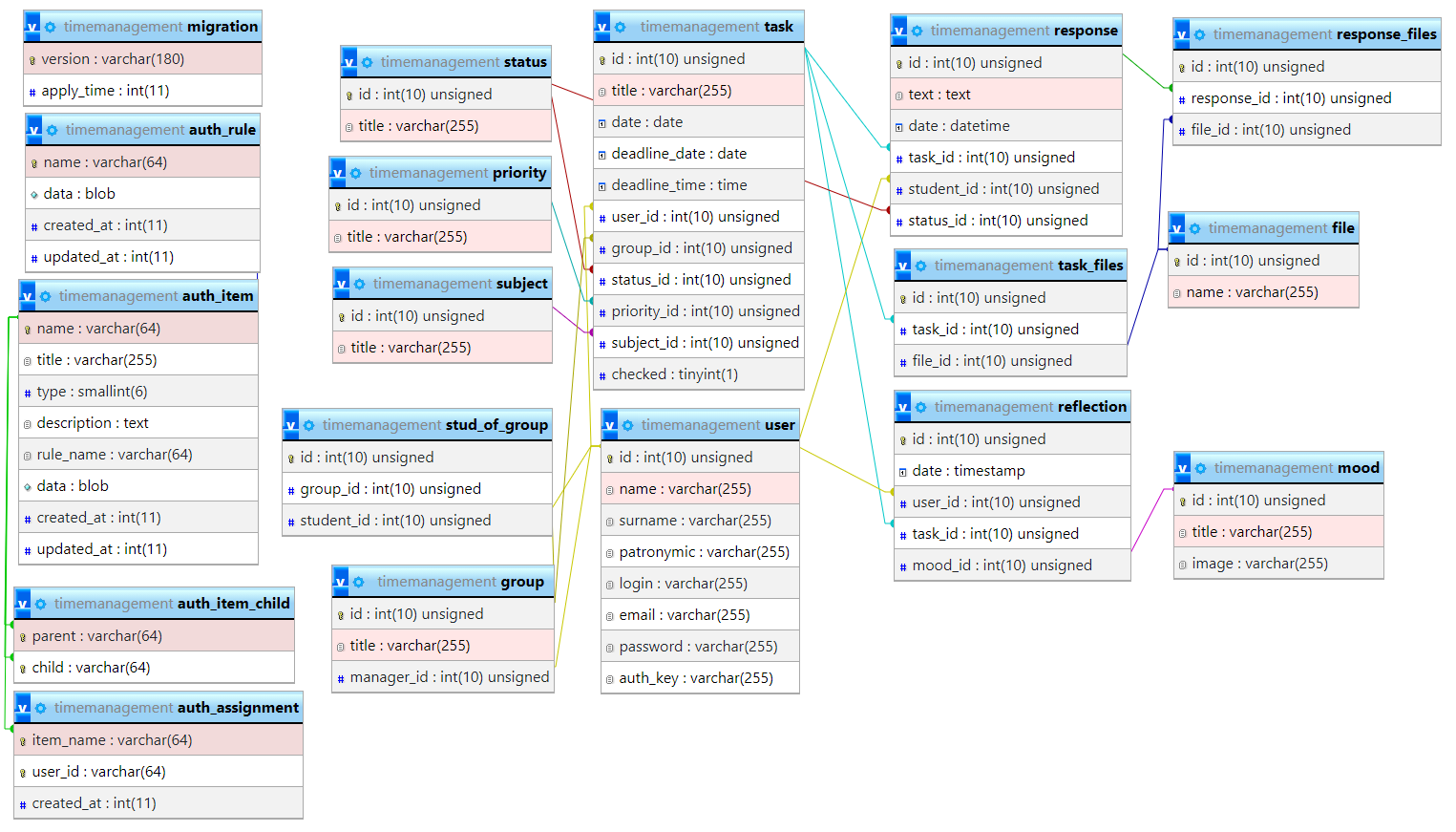


Рисунок 14 – База данных

2.6. Выбор средств реализации проекта.

Тщательно подойдя к выбору технологий для разработки моего проекта, составлено краткое описание преимуществ выбранных платформ.

Язык программирования Php:

* простота использования, простой синтаксис;
* легкость изучения (доступные учебные пособия) и читабельность кода;
* гибкость, т.е. способность смешиваться с другими языками;
* работа с фреймворком Yii2;
* универсальность (подходит для различных проектов).

Фреймворк Yii2:

* простой и быстрый (создание сайта из компонентов);
* функциональность (административная панель, формы, миграции и др.);
* безопасность;
* работа с крупными проектами и большой загруженностью;
* существует два заготовленных шаблона проекта для разработчиков с разным уровнем подготовки;
* имеет веб-инструмент для генерации кода, что обеспечивает быстроту его написания;
* имеет замечательную документацию, гайды по старту.

База данных MySQL:

* высокая скорость и производительность,
* простота использования,
* безопасность и простота установки,
* бесплатное обслуживание и открытый доступ.

HTML, CSS и JS фреймворк Bootstrap:

* высокая скорость создания сайта;
* корректная работа сайта во всех поддерживаемых браузерах и ос;
* готовые компоненты и шаблоны;
* элементарные навыки верстки;
* однородность дизайна;
* легкость изучения (огромное количество обучающих материалов).

Среда разработки Visual Studio Code:

* помощь при написании кода (функции автодополнения и анализа кода, подсветка и исправление ошибок);
* встроенные инструменты (инструмент запуска тестов, терминал, инструменты для работы с базами данных);
* поддержка различных веб-фреймворков.

Локальный сервер OpenServer:

* не требует установки на компьютер — его можно запустить с флешки или внешнего жёсткого диска;
* быстро запускается и останавливается;
* сервер автоматически начинает работу при запуске программы;
* можно выбрать режим управления доменами;
* возможна работа через командную строку;
* можно переключать модули http, mysql, php;
* достаточно одного клика для доступа к доменам;
* интерфейс поддерживает несколько языков.

## 3. ПОЭТАПНОЕ ОПИСАНИЕ РЕАЛИЗАЦИИ КОМПОНЕНТОВ ИС

3.1. Подготовка.

После развертывания проекта на сервере, в файле config/db.php указываются данные для подключения к базе данных – ее название, логин и пароль для авторизации.

В config/web.php необходимо указать различные параметры сайта:

* name – название, указываемое в шапке сайта;
* language – язык сайта;
* cookieValidationKey – ключ для проверки дополнительная безопасность проверки cookie, которая защищает файлы cookie от изменения на стороне клиента;
* baseUrl – базовый путь.

Также раскомментируется блок urlManager в конце файла для активации ЧПУ.

После успешного подключения БД создаются модели таблиц (рисунок 15).

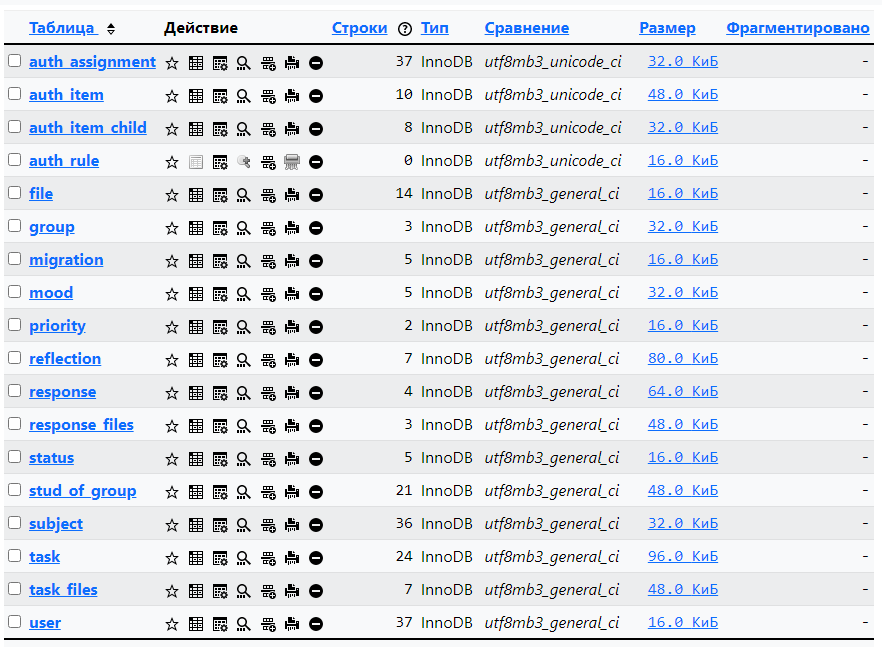


Рисунок 15 – Модели таблиц

Далее требуется применить шаблон к системе. Сначала css и js файлы перемещаются в отдельную папку в директории web. Прописывается вывод контента на шаблон через вывод переменной $content.

Для корректного импорта стилей и скриптов создаётся отдельный AssetBundle с названием AppAsset, как указано на рисунке 16. В нем указываются все стили и скрипты из шаблона.



Рисунок 16 – Класс AppAsset

3.2. Создание ролей с помощью RBAC.

RBAC (Role Based Access Control) в php-фреймворке Yii2 переводится как система доступа на основе ролей.

По сравнению с фильтрами контроля доступа ACF, которые делят всех посетителей сайта только на 2 категории - гости и авторизованные пользователи, RBAC гораздо более мощный инструмент. Он позволяет разделить авторизованных пользователей на неограниченное количество групп, давая каждой из них отдельные права.

Чтобы подключить систему, необходимо настроить компонент приложения authManager как в файле config/web.php, так и в конфигурации консольного приложения config/console.php (рисунок 17).

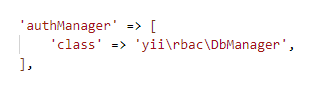


Рисунок 17 – Настройка DbManager

Прежде чем начать использовать этот менеджер, нужно создать необходимые таблицы в базе данных. Чтобы сделать это, в консоли прописывается следующая миграция:

yii migrate --migrationPath=@yii/rbac/migrations.

Теперь authManager может быть доступен через \Yii::$app->authManager.

RBAC позволяет создать роли автоматически с помощью консольного скрипта. Этот скрипт прописывается в файле RbacController.php. На стадии проектирования были определены основные роли пользователей. Таким образом, получились разграничения ролей, указанные на рисунке 18.

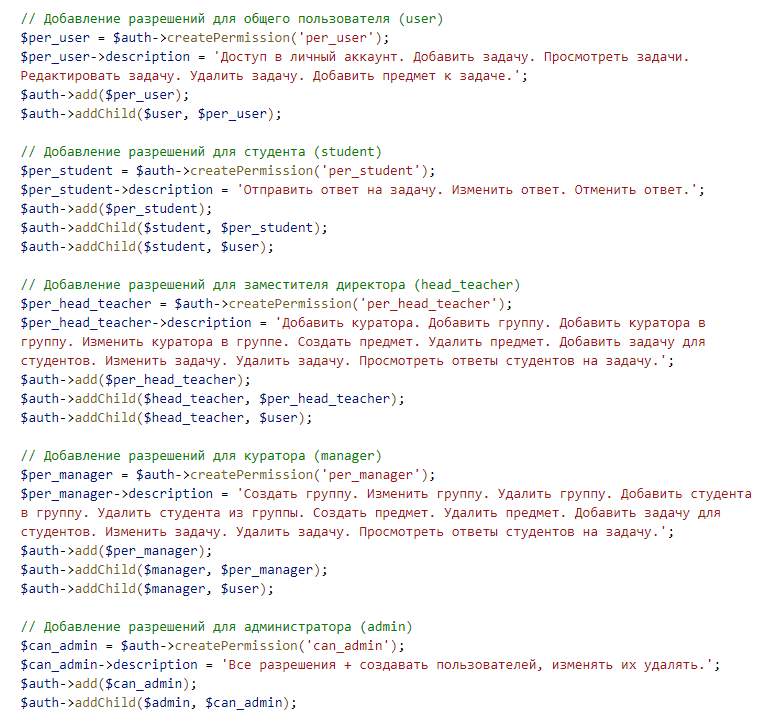


Рисунок 18 – RbacController

После выполнения команды yii rbac/init, в созданные ранее таблицы базы данных добавятся заполненные данные.

3.3. Страницы авторизации и регистрации.

Для начала, в классах RegisterForm и LoginForm прописываются свойства и правила валидации полей форм в соответствии с их назначением.

Создается и сохраняется модель пользователя с указанными данными в методе registerUser, код которого представлен на рисунке 19.



Рисунок 19 – Метод registerUser

Благодаря authManager в методе реализовано присвоение пользователю одной из ролей.

Нажав на ссылку регистрации или входа, пользователю открывается соответствующая страница, на которой предлагается заполнить поля, необходимые для того, чтобы зарегистрироваться в системе или же войти в уже существующий аккаунт. Этот механизм реализован с помошью методов actionLogin и actionRegister, которые принадлежат контроллеру SiteController.

В методе actionRegister (рисунок 20) происходит рендеринг страницы регистрации и вызов метода registerUser с последующей авторизацией и переадресацией на главную страницу.



Рисунок 20 – Метод actionRegister

На странице регистрации, пользователю необходимо заполнить обязательные поля. Для проверки данных без перезагрузки страницы устанавливается свойство enableAjaxValidation для полей email и login. Форма регистрации показана на рисунке 21.

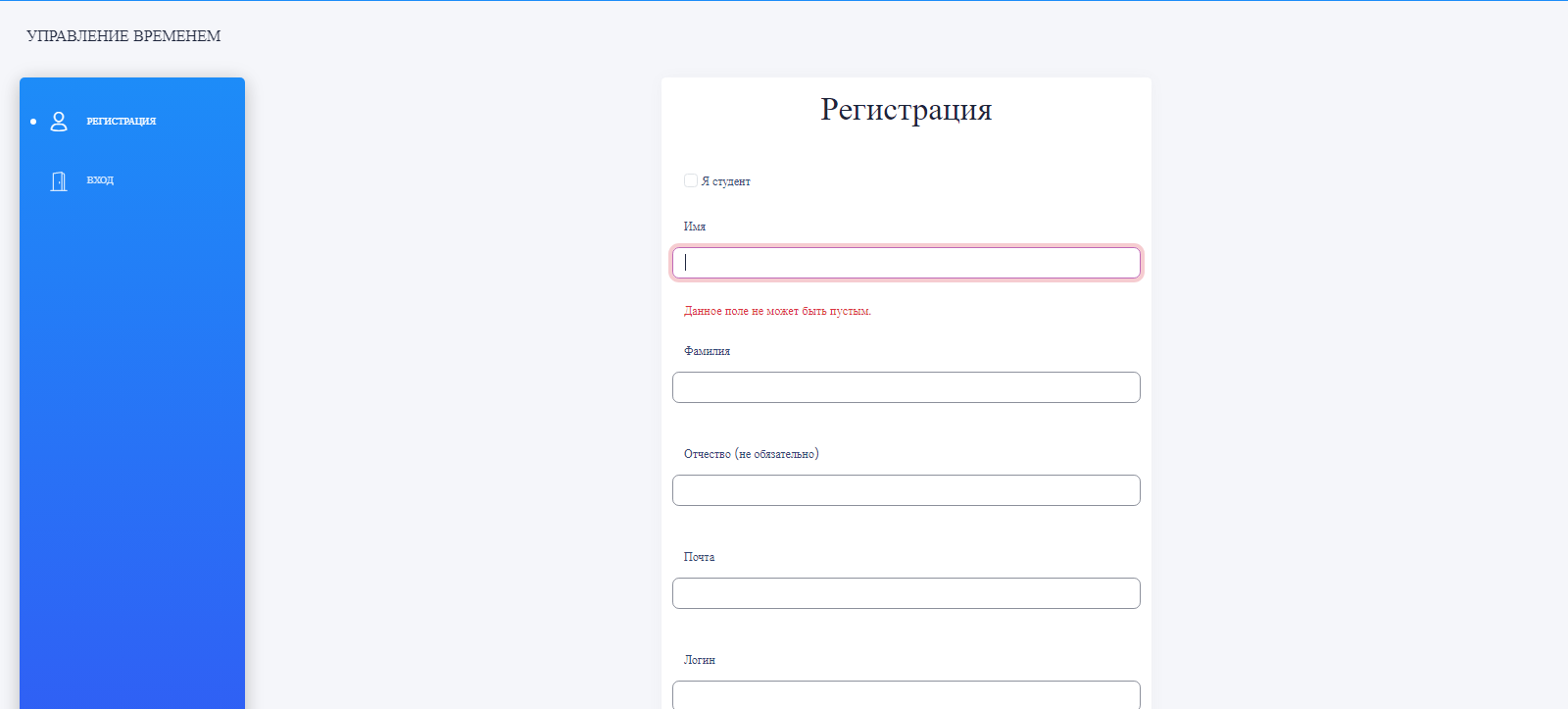


Рисунок 21 – Страница регистрации

Класс User дорабатывается для использования интерфейса IdentityInterface. В него включаются все требуемые методы из документации, а также добавляются следующие методы:

* beforeSave – создание хэша пароля, ключа аутентификации, установка роли, непосредственно перед сохранением модели;
* findByUsername – поиск экземпляра класса User по его логину с помощью статического метода findOne;
* validatePassword – сравнение пароля с его хэшем в базе данных.

На странице входа пользователю необходимо заполнить логин и пароль, чтобы войти в систему (рисунок 22).

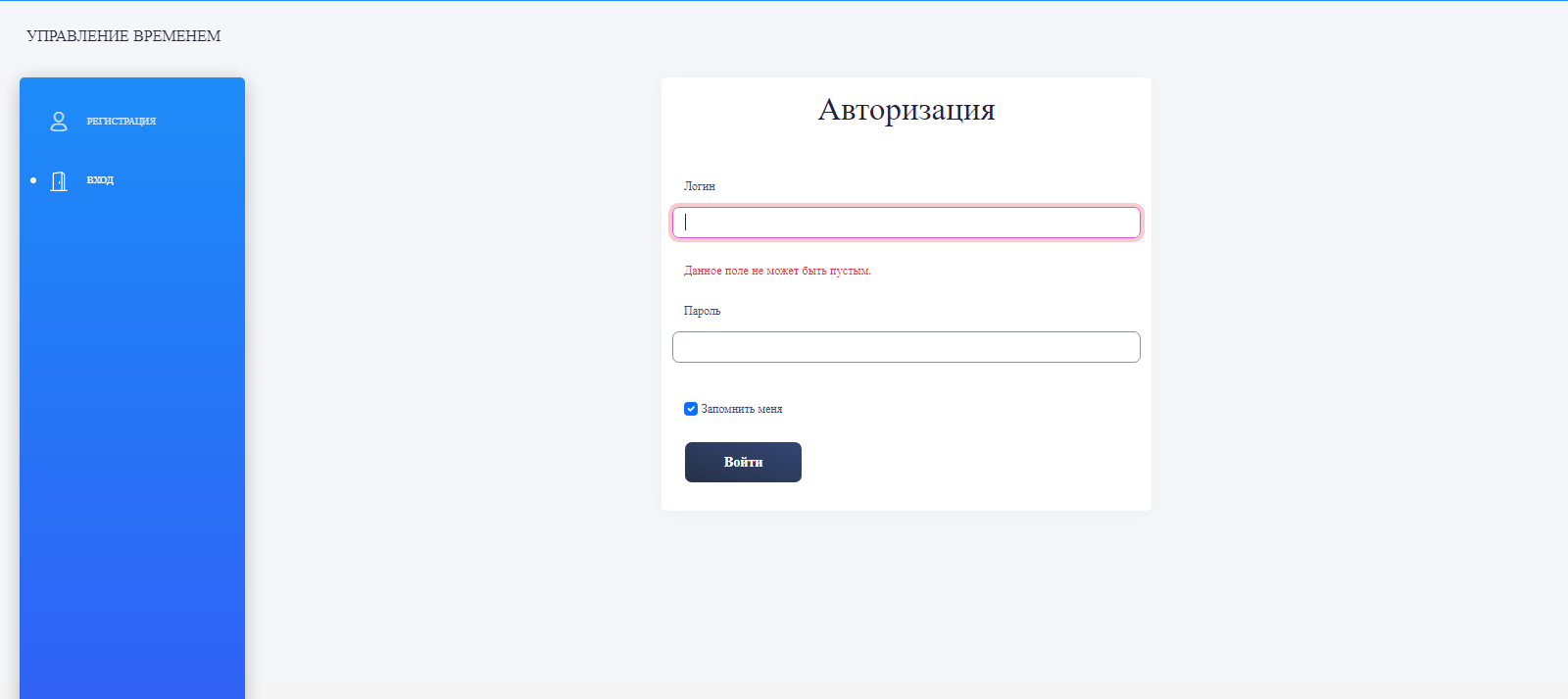


Рисунок 22 – Страница входа

3.4. Главная страница и личный кабинет.

Навигационная панель неавторизованного пользователя включает в себя ссылки на страницу регистрации и страницу входа для зарегистрированного пользователя (рисунок 23).

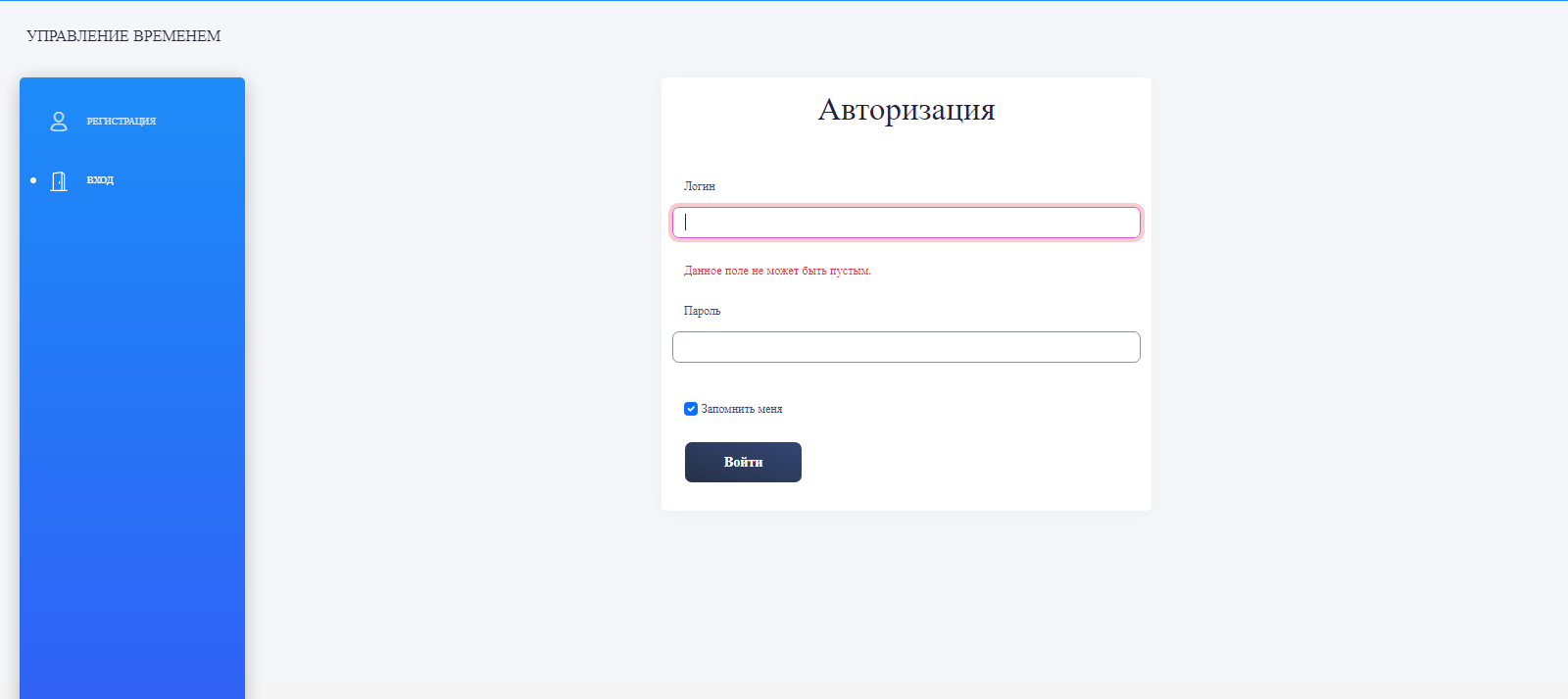


Рисунок 23 – Навигационная панель сайта

После авторизации, пользователь попадает на главную страницу, которая включает в себя блок с приветствием пользователя, а также кнопки для перехода на доступные ему страницы. Главная страница изображена на рисунке 24.

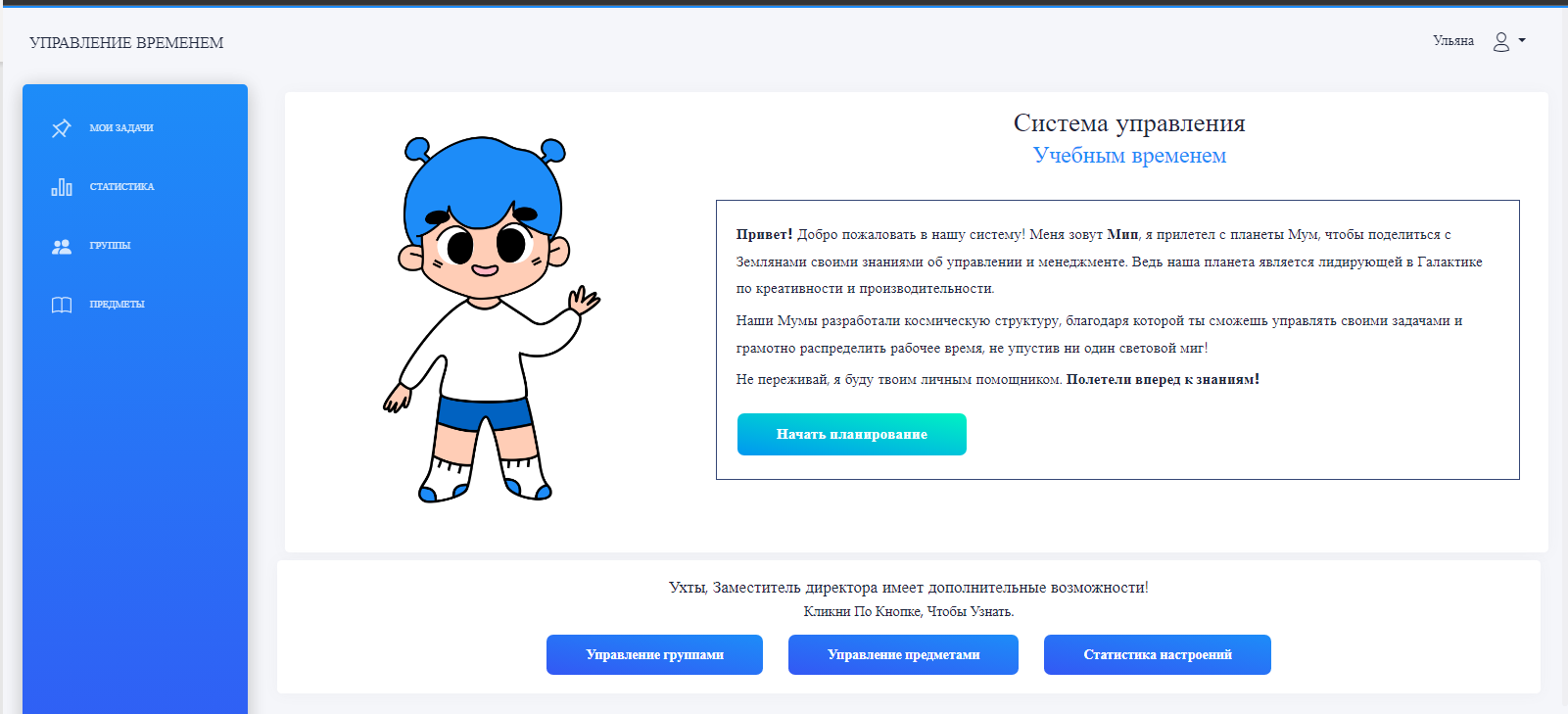


Рисунок 24 – Главная страница сайта

Чтобы пользователь мог просматривать и изменять личную информацию, был создан контроллер ProfileController в модуле account. В личном кабинете есть возможность изменить имя, фамилию, отчество, почту и логин пользователя. Также, есть страница смены пароля, на которой проверяется соответствие старому паролю, и при успешной проверке создается новый. Внешний вид личного кабинета показан на рисунке 25.

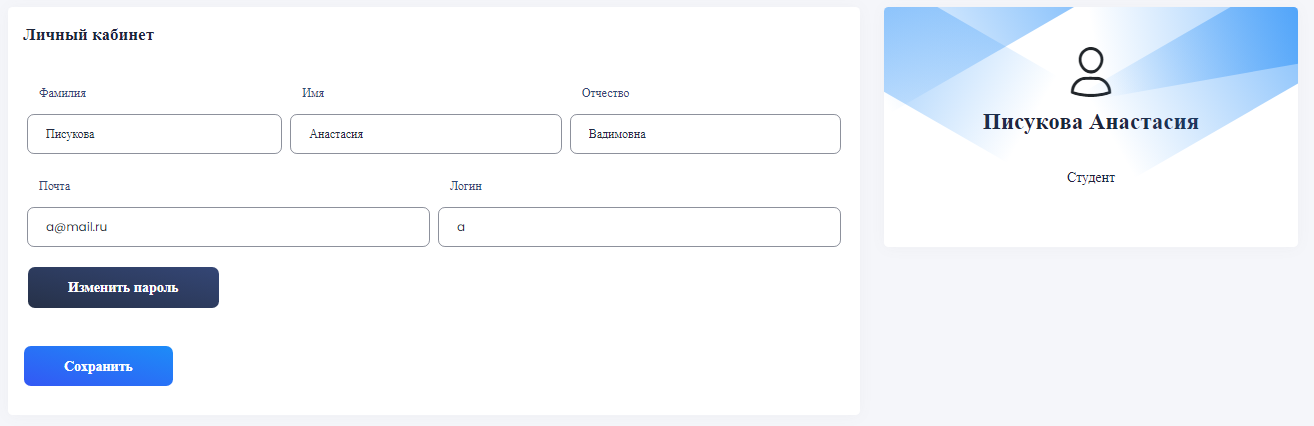


Рисунок 25 – Личный кабинет

3.5. Управление задачами.

Самой основной функциональной единицей проекта является управление задачами. Для того, чтобы реализовать ее, создаётся модуль account, и на основе него создается контроллер TaskController. В контроллере реализуется следующий алгоритм управления задачами:

* создание задачи,
* вывод списка задач для каждого пользователя по данным из БД,
* просмотр данных задачи,
* редактирование задачи,
* выполнение задачи,
* удаление задачи,
* сортировка, фильтрация и поиск по задачам.

Текущему пользователю выводятся только его задачи на экран в виде отдельных строк. В каждой строке созданной пользователем задачи

Чтобы найти нужную задачу, на странице представлены 5 различных вариантов фильтрации: поиск по названию задачи, выборка по степени важности, по статусу и по академическому предмету, а также календарь для просмотра задач по сроку их выполнения. Календарь реализован с помощью виджета DatePicker, импортированного с сайта с готовыми шаблонами виджетов даты и времени krajee. Программный код фильтрации задач отображен на рисунке 26.

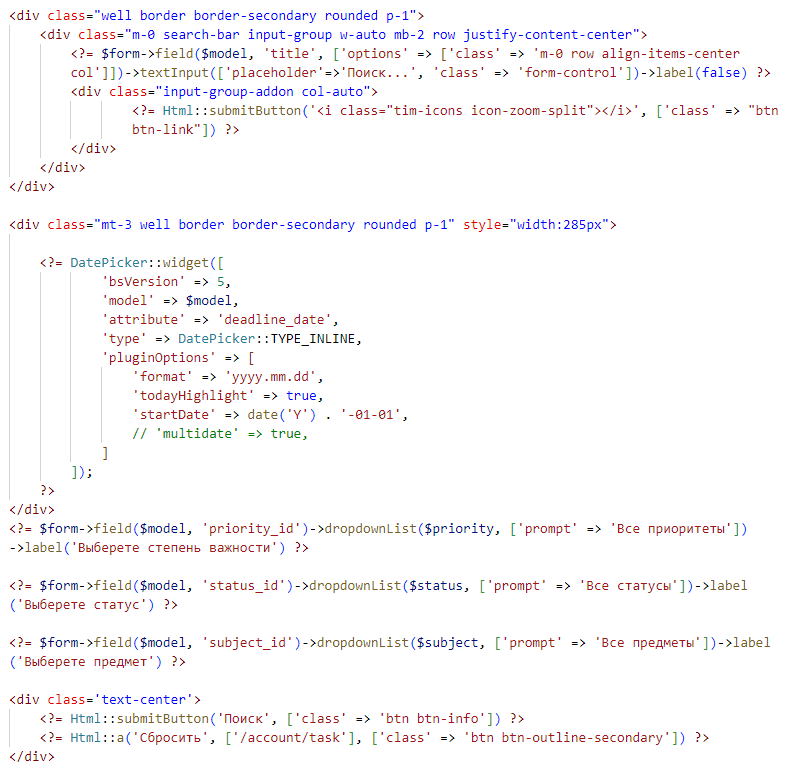


Рисунок 26 – Фильтрация задач

Для того, чтобы реализовать выполнение задачи при нажатии на чек-бокс, необходимо было включить в проект JavaScript. Запрос к серверу без перезагрузки страницы методом POST осуществляет jQuery.post(). Чтобы эта функция начала работать, был создан файл checkbox.js и в нем прописан соответствующий программный код, который показан на рисунке 27.

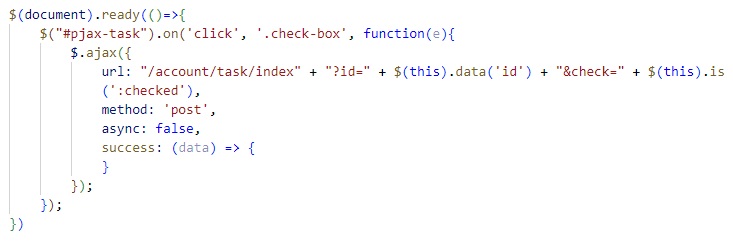


Рисунок 27 – Checkbox.js

Контроллер TaskController перехватывает данные из созданного POST запроса и с их помощью статус задачи меняется на «Выполнено». В приложении А представлен полный код алгоритма.

Внешний вид готовой панели вместе с описанным блоком фильтрации представлен на рисунке 28.

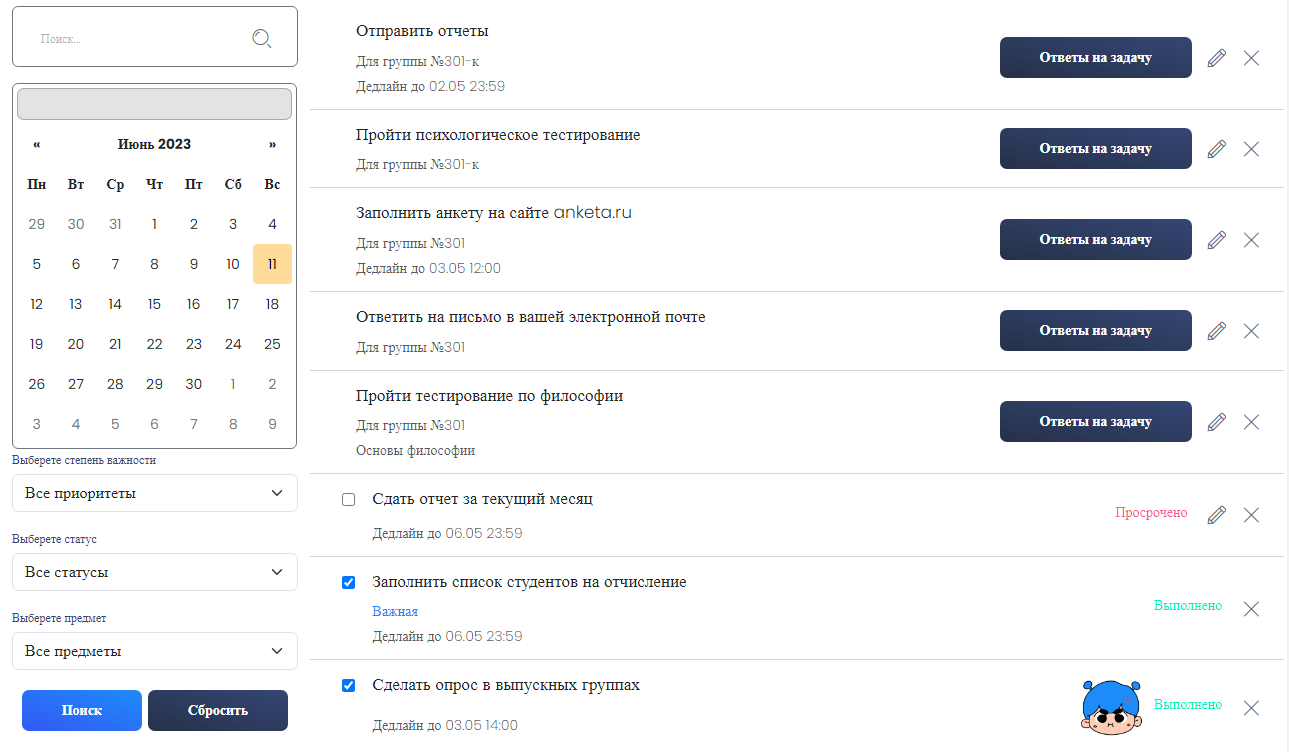


Рисунок 28 – Панель управления задачами

При создании задачи были осуществлены возможности установки даты и времени дедлайна через krajee DatePicker, а также возможности прикрепления файлов к задаче с помощью механизма загрузки нескольких файлов в Yii2. Чтобы у пользователя была возможность управлять прикрепленными к задаче файлами, на странице редактирования задачи был добавлен функционал удаления файла. В приложении А представлен полный код редактирования задачи. На рисунке 29 можно увидеть внешний вид страницы редактирования.

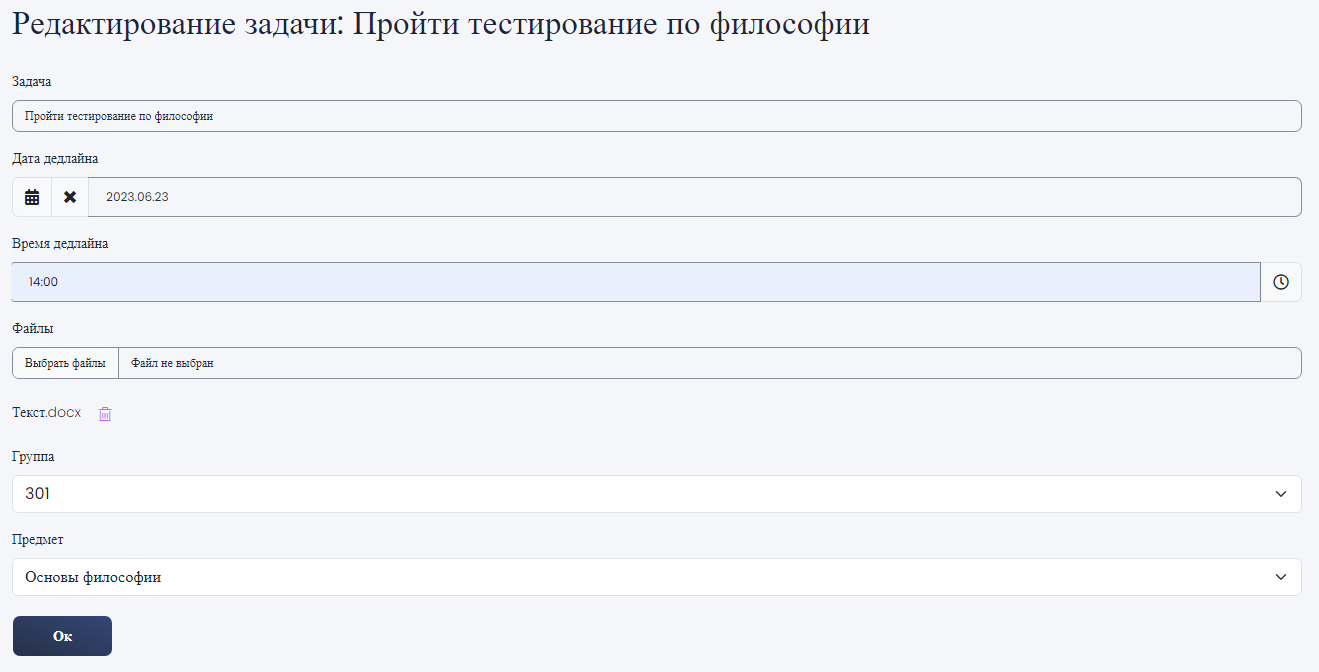


Рисунок 29 – Редактирование задачи

Для того, чтобы пользователю была доступна оценка выполнения своих задач, был создан ReflectionController в модуле account. При выполнении задачи появляется модальное окно, в котором предлагается выбрать эмоцию, испытываемую пользователем на данный момент. Как только эмоция была выбрана, она сохраняется в базе данных и отображается рядом с задачей. Внешний вид модального окна представлен на рисунке 30.



Рисунок 30 – Модальное окно с выбором эмоций

После первой отмеченной эмоции пользователю доступна страница со статистикой его рефлексии. На странице отображена столбчатая диаграмма, на которой показывается количество отмеченных пользователем эмоций, чтобы он в дальнейшем смог сделать анализ своего эмоционального состояния во время выполнения работы. График был создан с использованием виджета highcharts на Yii2. Внешний вид страницы можно увидеть на рисунке 31.



Рисунок 31 – Статистика рефлексии

3.6. Управление ответами на задачи.

Если куратор или заместитель директора создали задачу непосредственно для группы, студент, входящий в ее состав, может отправить свой ответ. Для этого создаётся контроллер ResponseController в модуле account. В контроллере реализуется следующий алгоритм управления ответами на задачи:

* создание ответа на задачу,
* просмотр ответа на задачу для пользователя по данным из БД,
* редактирование ответа,
* удаление ответа,
* просмотр списка ответов студентов на конкретную задачу.

При просмотре задачи студенту выводится кнопка «Прикрепить ответ», если ответ еще не создан. В противном случае, выводится кнопка, по которой можно просмотреть данные своего ответа, а также отредактировать или удалить его. На странице редактирования ответа реализован функционал управления файлами. Внешний вид страницы представлен на рисунке 32.

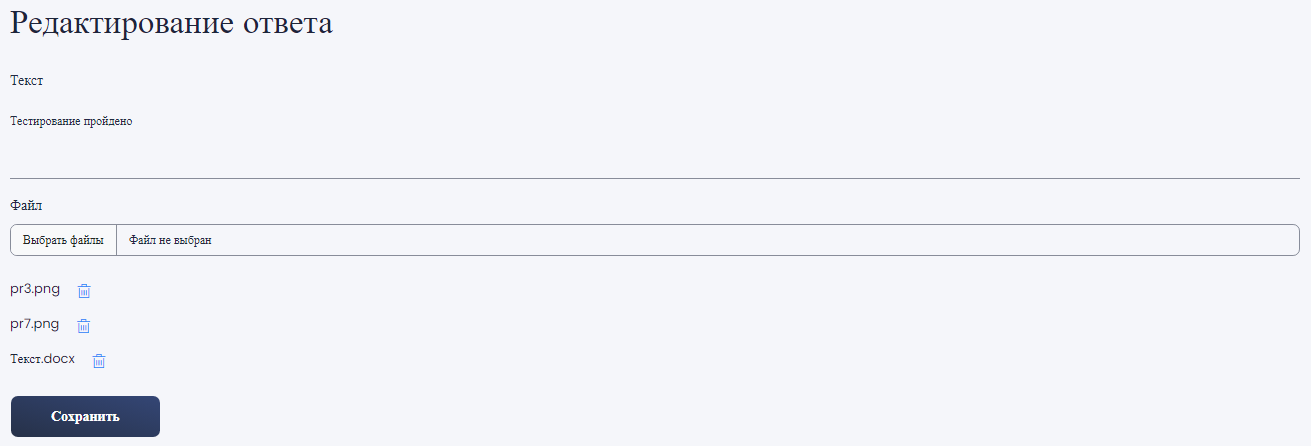


Рисунок 32 – Редактирование ответа

Отправитель задачи может посмотреть список отправленных студентами ответов, в дополнение к этому есть возможность отфильтровать их по статусу и просмотреть список студентов, которые не прикрепили ответ. Внешний вид готовой панели представлен на рисунке 33.

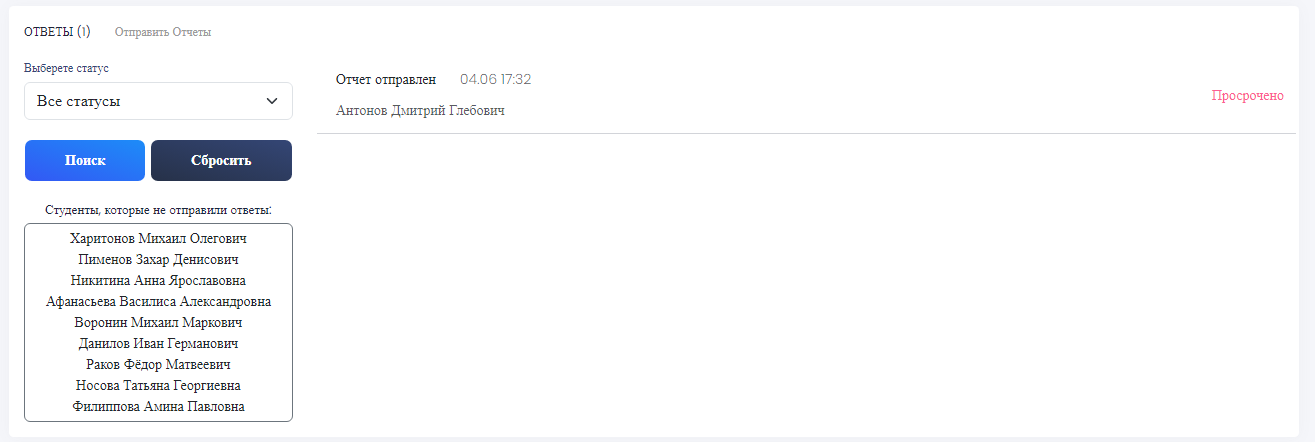


Рисунок 33 – Ответы на задачу

3.7. Управление группами.

В модуле account создаётся контроллер GroupController. В контроллере реализуется следующий алгоритм управления группами:

* создание группы,
* просмотр группы по данным из БД,
* редактирование группы,
* удаление группы,
* управление студентами в группе.

Последний алгоритм доступен только роли куратора, так как именно он имеет право добавлять студентов в группу и удалять студентов из группы. Для этого был создан еще один контроллер StudOfGroupController на основе модели StudOfGroup, хранящей связи студентов и групп. На реализованной странице куратор может выбрать из выпадающего списка студентов, еще не прикрепленных к группе, и добавить их в свою группу. Внешний вид панели изображен на рисунке 34.

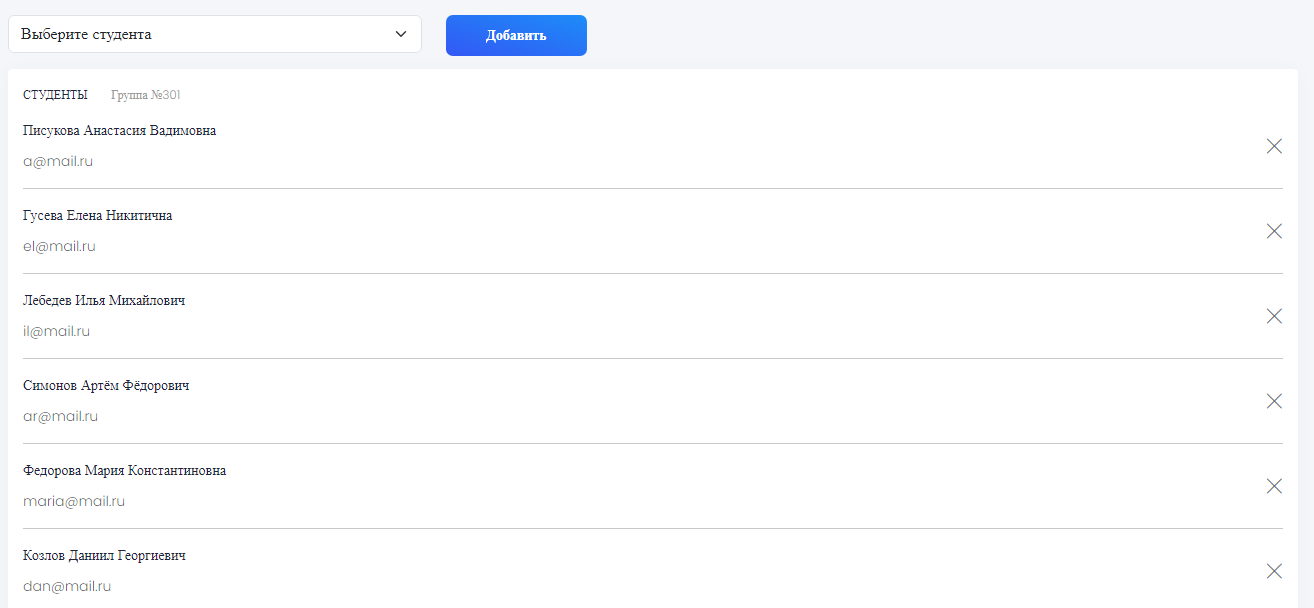


Рисунок 34 – Управление студентами в группе

3.8. Управление академическими предметами.

Функционал управления предметами доступен только роли заместителя директора и администратора. В модуле account создаётся контроллер SubjectController. В контроллере реализуется следующий алгоритм управления предметами:

* создание предмета,
* просмотр предметов по данным из БД,
* редактирование предмета,
* удаление предмета,

Можно осуществлять поиск по названию академических предметов с помощью поля поиска в верхней части страницы. Внешний вид панели изображен на рисунке 35.

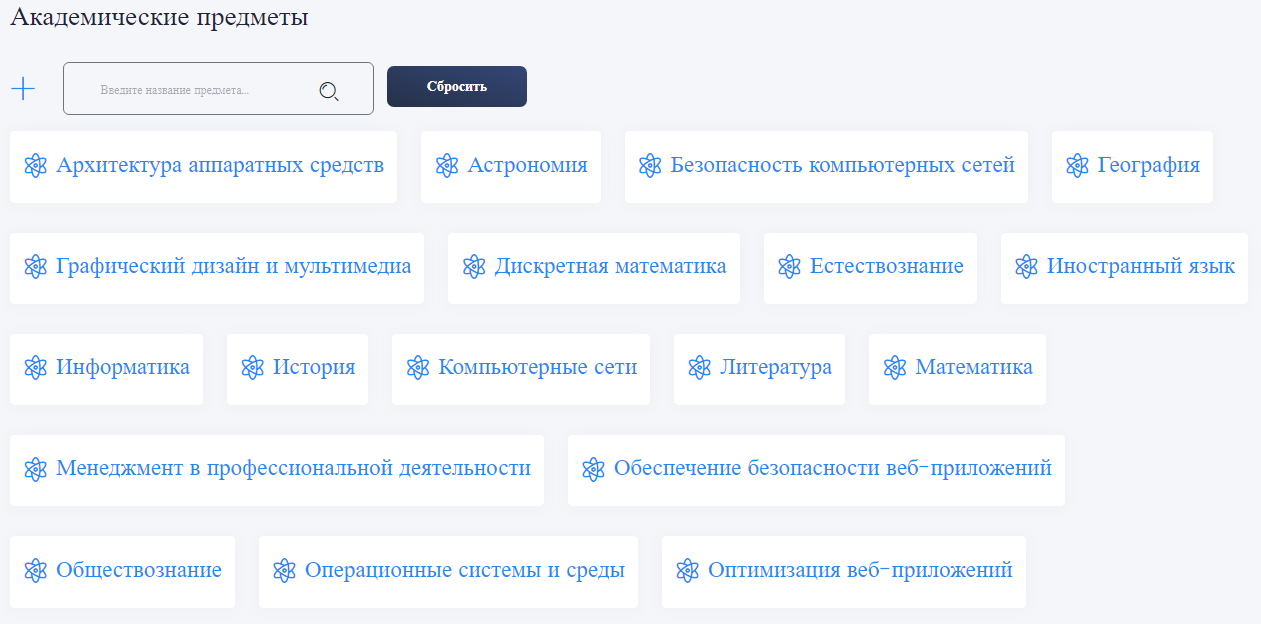


Рисунок 35 – Управление академическими предметами

3.9. Панель администрирования.

С помощью генератора Gii создаётся модуль admin. Далее генерируются CRUD для модели User. На основе формы регистрации создается форма UserCreateForm для создания пользователя. При помощи переменной «role» и метода getRoleList в модели AuthItem реализован выпадающий список с доступными для присвоения ролями. Внешний вид панели изображен на рисунке 36.

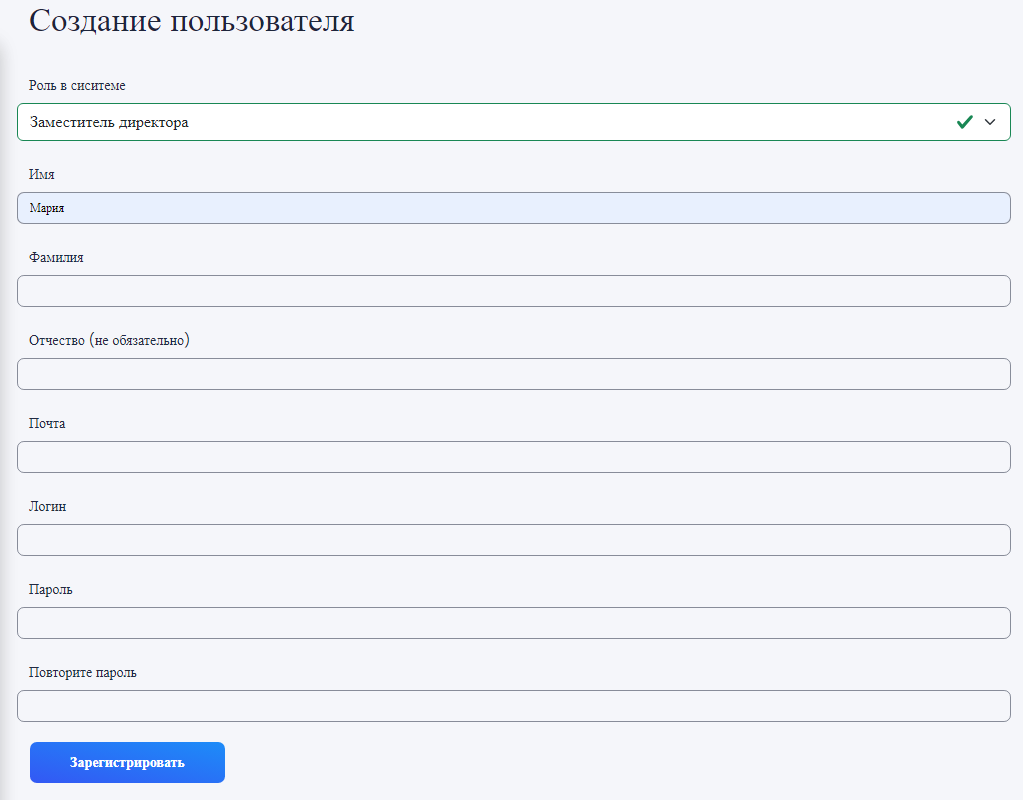


Рисунок 36 – Управление пользователями

3.10. Руководство пользователя.

**Общее описание системы.**

Информационная система управления учебным временем студента является удобным инструментом для управления задачами с возможностью их выполнения, позволяющим не только прикреплять ответы к задачам для группы, но и работать с рефлексией и видеть статистику по ней.

**Возможности системы.**

Всех пользователей системы можно разделить на 2 роли: пользователь и администратор. Роль пользователя делится на роли студент, куратор и заместитель директора. Администратор, в свою очередь, является все тем же обычным пользователем, но имеет доступ к управлению пользователями, что расширяет его возможности.

**Руководство пользователя: «Пользователь».**

1. Регистрация и авторизация в системе.

Для того, чтобы авторизоваться или зарегистрироваться, пользователю необходимо нажать на соответствующую кнопку в навигационной панели сайта, которая изображена на рисунке 37.

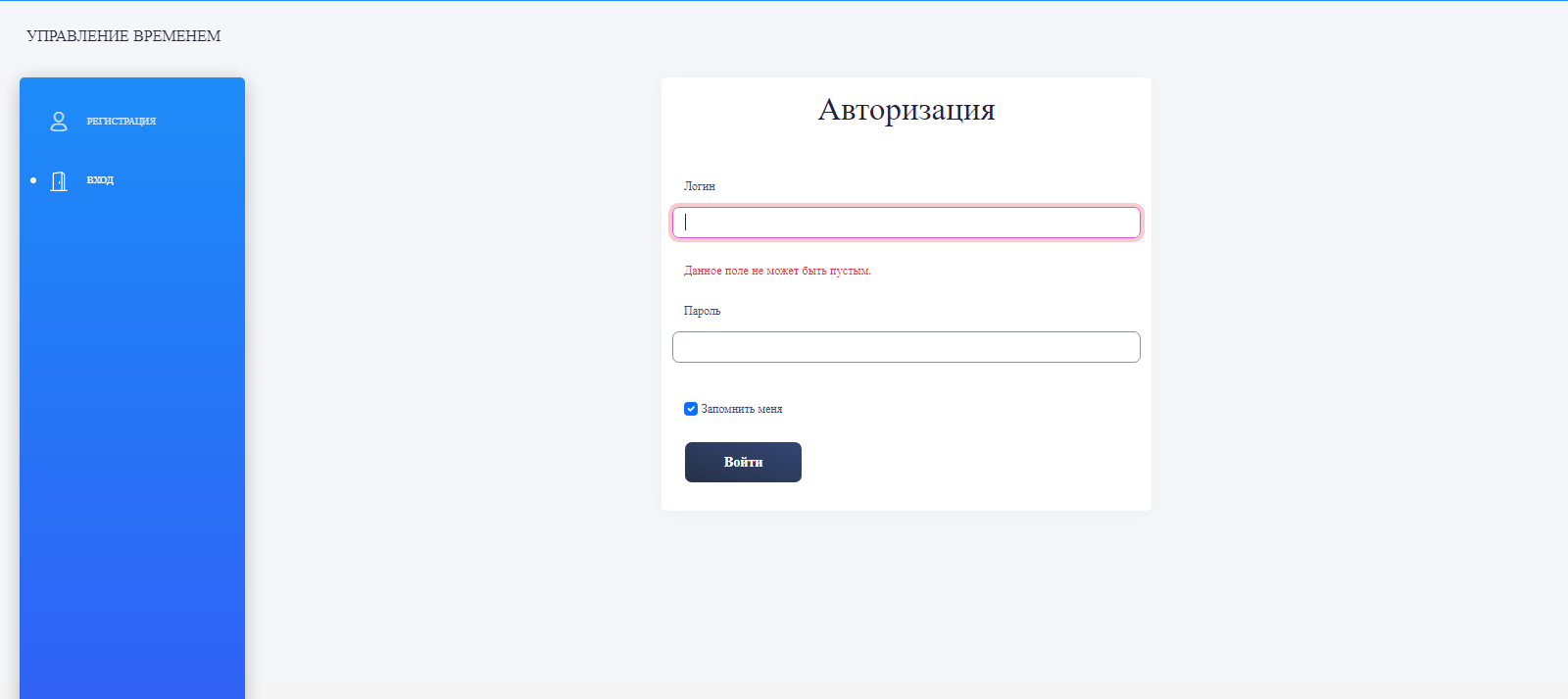


Рисунок 37 – Кнопки входа и регистрации.

После нажатия на одну из кнопок, откроется соответствующая страница, предлагающая ввести данные, необходимые для выполнения выбранного действия. Чтобы зарегистрироваться, как студент, нужно нажать галочку в поле «Я студент», в противном случае пользователь будет зарегистрирован под ролью куратора.

1. Создание задачи.

Для создания новой задачи пользователю нужно перейти на страницу «Мои задачи» и нажать на большую кнопку в виде математического знака плюс, находящуюся вначале страницы. После нажатия, откроется страница с формой создания задачи, где пользователю необходимо ввести данные: текст задачи и дополнительная информация, такая как дата и время дедлайна, файлы, академический предмет, приоритет. Информация о задаче будет отображаться на странице с задачами, впоследствии ее можно будет изменить.

3. Выполнение задачи.

Для выполнения задачи пользователь должен перейти на страницу с задачами. После этого, необходимо нажать на чек-бокс той задачи, которую он желает выполнить, чтобы поставить галочку. После нажатия на чек-бокс, откроется модальное окно рефлексии, показанное на рисунке 38. Выбрав подходящую эмоцию, пользователю необходимо нажать на кнопку «Ок». После отправления, ответ будет записан в базу данных, а пользователю откроется страница с задачами.



Рисунок 38 – Модальное окно

4. Управление личным кабинетом.

Для изменения личных данных пользователя, ему нужно перейти в личный кабинет, ссылка на который находится в выпадающем списке в правом верхнем углу экрана (рисунок 39).

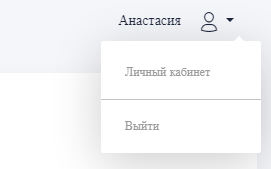


Рисунок 39 – Ссылка на личный кабинет

Далее, можно поменять информацию в любом из отображенных полей, и нажать кнопку «Сохранить». Чтобы изменить пароль, нужно нажать на кнопку «Изменить пароль», ввести в поля необходимые данные (старый и новый пароль) и нажать кнопку «Сменить пароль». При успешном выполнении любой из операций, появится сообщение, представленное на рисунке 40.

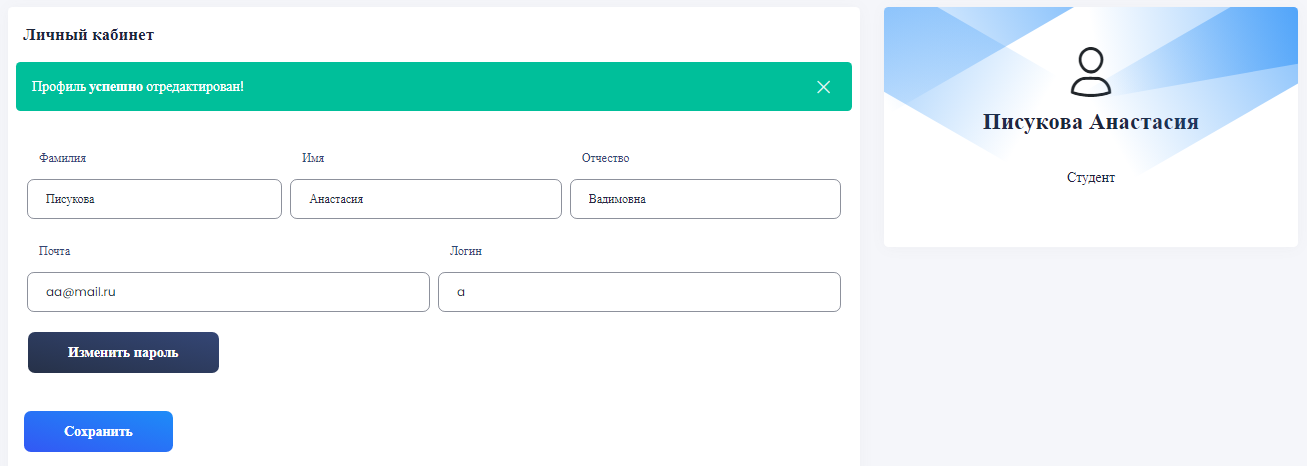


Рисунок 40 – Успешное редактирование профиля

**Руководство пользователя: «Студент».**

1. Создание ответа на задачу для группы.

Чтобы прикрепить к задаче для группы ответ, необходимо нажать на кнопку «Прикрепить ответ» рядом с нужной задачей и перейти на страницу создания ответа, изображенной на рисунке 41. После ввода текста ответа и прикрепления файла (при необходимости), нужно нажать кнопку «Сохранить», чтобы данные сохранись в системе. В дальнейшем, созданный ответ можно просмотреть, удалить или отредактировать (если у него нет статуса «Проверено»), перейдя по кнопке «Ответ на задачу», расположенной рядом с задачей.

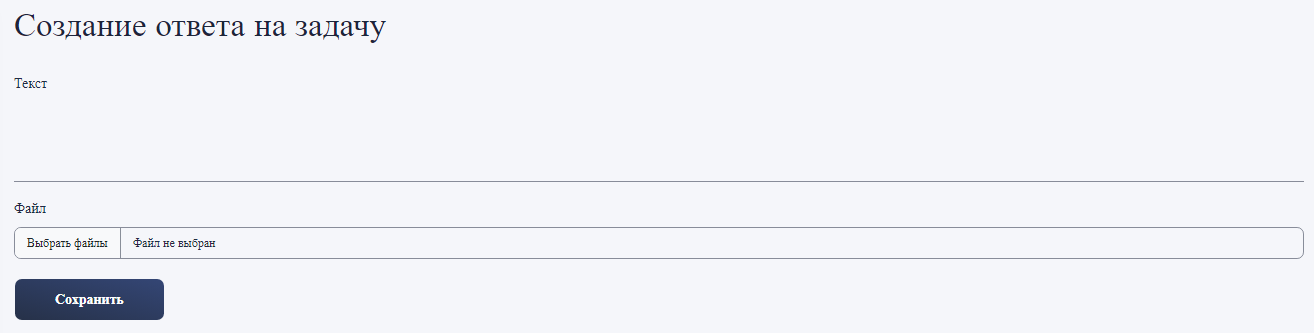


Рисунок 41 – Создание ответа на задачу

**Руководство пользователя: «Куратор».**

1. Проверка ответов на задачу для группы.

Чтобы просмотреть список студентов, прикрепивших ответ к задаче для группы, необходимо нажать на кнопку «Ответы на задачу» рядом с нужной задачей и перейти на страницу ответов, изображенной на рисунке 42. Нажав на текст нужного ответа, произойдет переход на страницу ответа с его информацией. Чтобы проверить ответ, нужно нажать на кнопку «Проверить». После этого, появится надпись «Проверено», а также кнопка «Перепроверить», по которой можно отменить предыдущее действие.

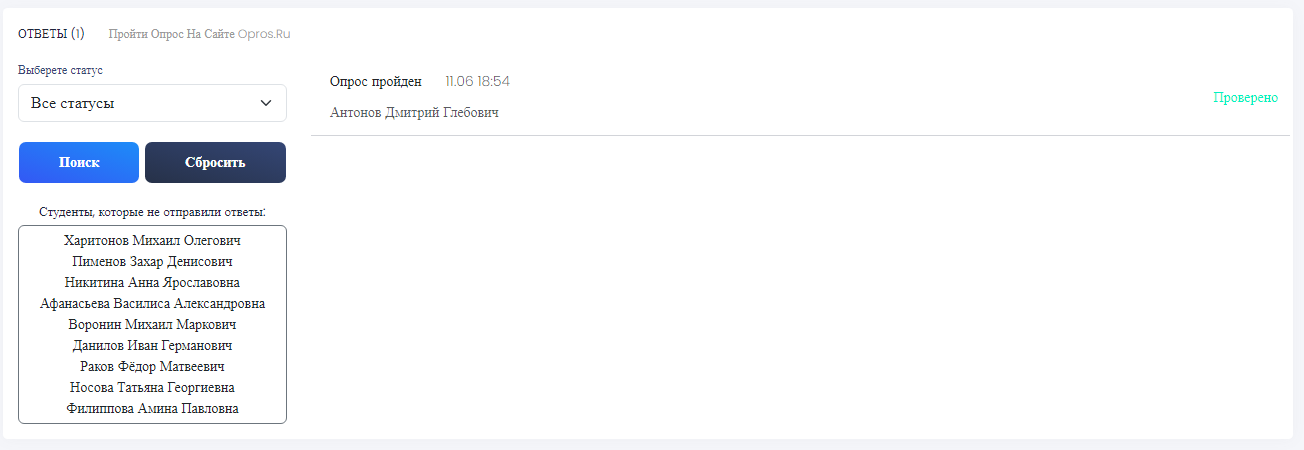


Рисунок 42 – Список ответов на задачу

1. Управление группой.

Куратор имеет возможность управлять студентами своих групп. Для этого нужно перейти на страницу «Мои группы» через навигационную панель, затем нажать на кнопку с номером нужной группы и перейти по кнопке «Управление студентами». Пользователю откроется страница с полем, в котором можно выбрать студента и, нажав на кнопку «Добавить», зачислить его в группу. Также, рядом с каждым студентом в списке есть кнопка в виде креста, которая позволяет удалить студента из группы. Внешний вид страницы представлен на рисунке 43.

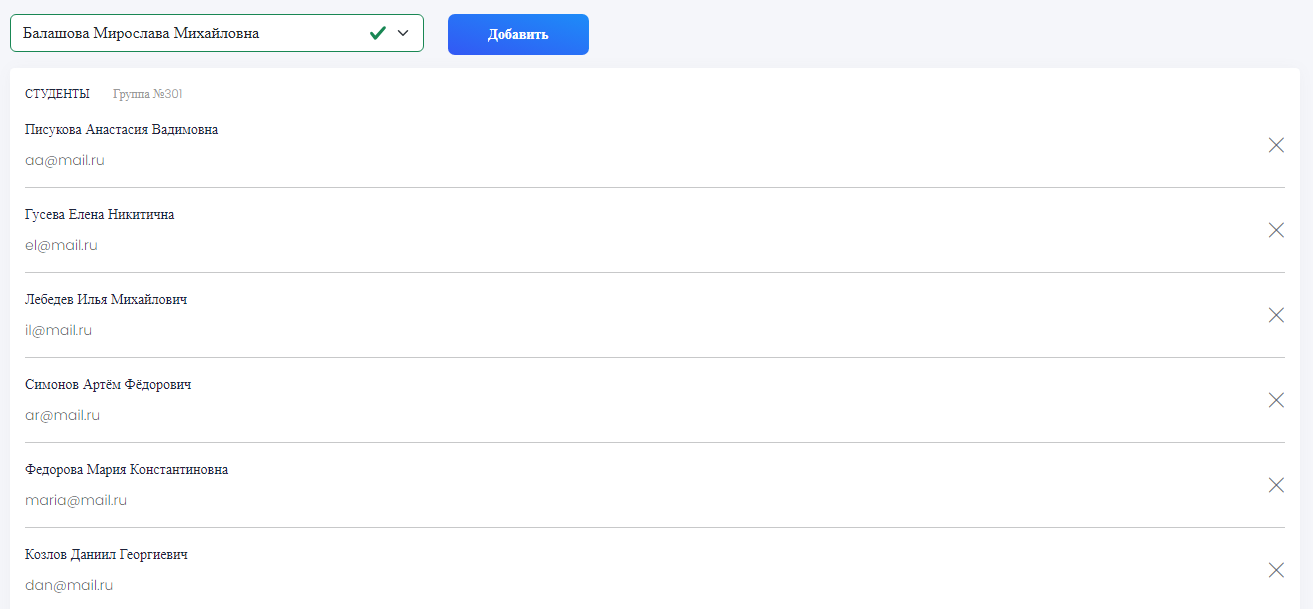


Рисунок 43 – Управление студентами

**Руководство пользователя: «Заместитель директора»**

1. Проверка ответов на задачу для группы.

Чтобы просмотреть список студентов, прикрепивших ответ к задаче для группы, необходимо нажать на кнопку «Ответы на задачу» рядом с нужной задачей и перейти на страницу ответов, изображенной на рисунке 44. Нажав на текст нужного ответа, произойдет переход на страницу ответа с его информацией. Чтобы проверить ответ, нужно нажать на кнопку «Проверить». После этого, появится надпись «Проверено», а также кнопка «Перепроверить», по которой можно отменить предыдущее действие.

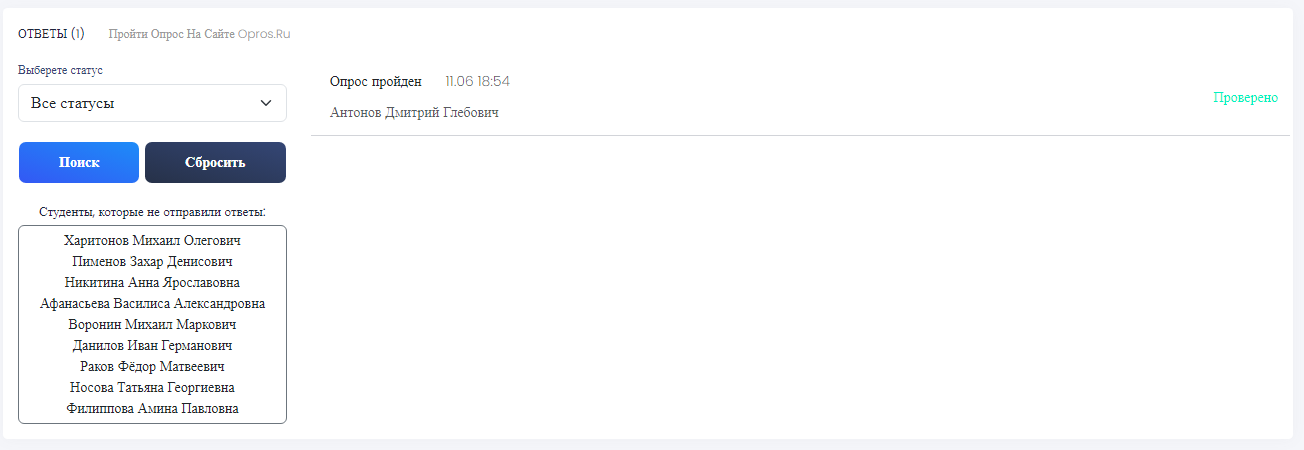


Рисунок 44 – Список ответов на задачу

1. Создание группы.

Чтобы создать группу, необходимо перейти на страницу «Группы» через ссылку в навигационной панели, затем нажать на большую кнопку в виде математического знака плюс, находящуюся вначале страницы. После нажатия, откроется страница с формой создания группы, где необходимо ввести номер группы и выбрать куратора из выпадающего списка. После сохранения группу можно будет отредактировать.

1. Создание академического предмета.

Для создания предмета нужно перейти на страницу «Предметы» через ссылку в навигационной панели, затем нажать на большую кнопку в виде математического знака плюс. После нажатия, откроется страница с формой создания предмета, где необходимо ввести его название. После сохранения предмет будет отображаться в списке всех предметов, его можно будет просмотреть и отредактировать.

**Руководство пользователя: «Администратор»**

1. Создание пользователя.

Для управления пользователями необходимо перейти на страницу «Пользователи» в навигационной панели. Далее нужно нажать на большую кнопку в виде математического знака плюс, после чего откроется страница с формой создания пользователя, где необходимо ввести его данные, а также выбрать роль из выпадающего списка. Поля формы отображены на рисунке 45. После сохранения пользователь будет отображаться в списке, его можно будет просмотреть, отредактировать и удалить.

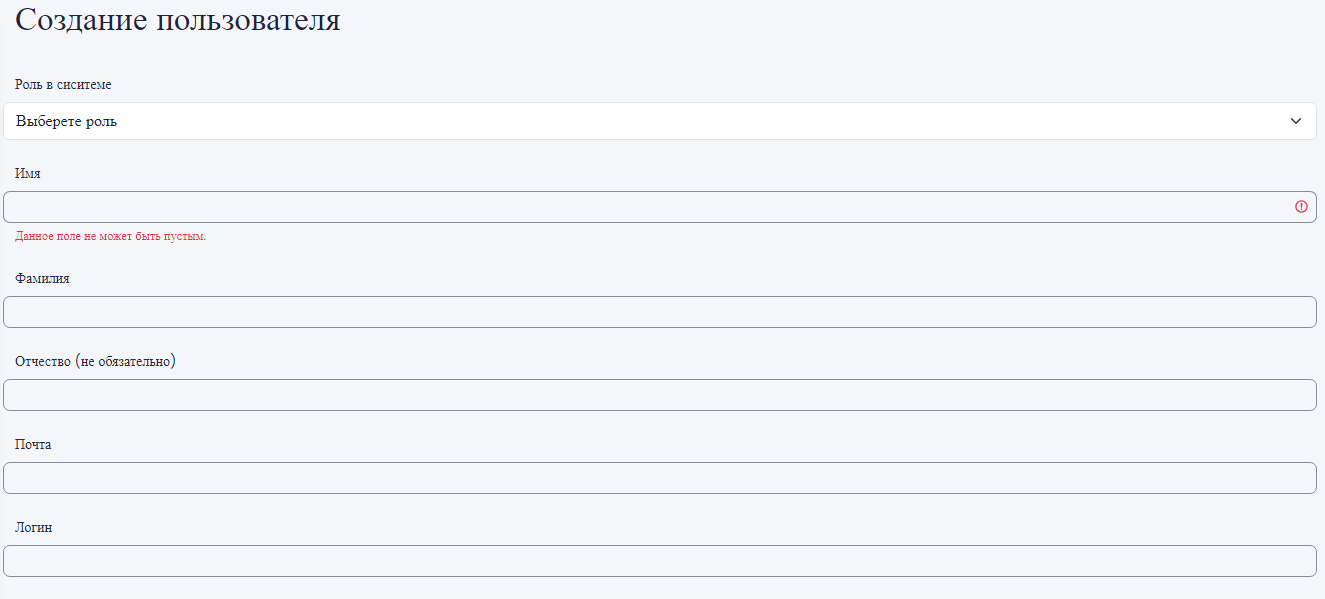


Рисунок 45 – Создание пользователя

1. Создание группы.

Чтобы создать группу, необходимо перейти на страницу «Группы» через ссылку в навигационной панели, затем нажать на большую кнопку в виде математического знака плюс, находящуюся вначале страницы. После нажатия, откроется страница с формой создания группы, где необходимо ввести номер группы и выбрать куратора из выпадающего списка. После сохранения группу можно будет отредактировать.

1. Создание академического предмета.

Для создания предмета нужно перейти на страницу «Предметы» через ссылку в навигационной панели, затем нажать на большую кнопку в виде математического знака плюс. После нажатия, откроется страница с формой создания предмета, где необходимо ввести его название. После сохранения предмет будет отображаться в списке всех предметов, его можно будет просмотреть и отредактировать.

3.11. Руководство программиста

Для корректной работы информационной системы, ее необходимо загрузить на хостинг.

После загрузки файлов на хостинг, нужно настроить базу данных, для этого необходимо создать новую базу данных MySQL на хостинге и импортировать имеющуюся базу, если база отсутствует – создать базу, соответствующую структуре ER-диаграммы.

Для подключения базы данных необходимо редактировать файл db.php, находящийся в папке проекта config. Необходимо изменить «username», «password» и адрес хоста, эти данные получаются при создании базы данных на хостинге.

Также, для корректной работы сервиса необходимо настроить .htaccess файл в корневой папке проекта. Для этого нужно ввести код, показанный на рисунке 46.



Рисунок 46 – Изменение файла .htaccess.

## ЭКОНОМИЧЕСКАЯ ЧАСТЬ

4.1. Определение затрат:

Определяют статьи затрат и нормы расхода по ним для последующего включения в расчет себестоимости программного продукта.

* *на вспомогательные материалы*, включающие: тонер для картриджа, бумага для принтера, флешка Kingston DataTraveler 100 G3 16 ГБ и т.п. (таблица 1);
* *на электроэнергию*, учитывая мощность потребляемой энергии компьютера, принтера и т.п. устройств;
* *на основную и дополнительную заработную плату* разработчика программного продукта;
* отчисления на социальные нужды;
* накладные расходы (укрупнено);

4.2. Расчет себестоимости программного продукта

Себестоимость программного продукта рассчитывается по формуле:

*С = Мвс + Э + Зозп + Здзп + Зсн+Н* ,

где - затраты на вспомогательные материалы, р.;

- затраты на электроэнергию на технологические цели, р.;

- основная зарплата разработчика, р.;

 *-* дополнительная зарплата разработчика, р.;

- взносы на социальное страхование и обеспечение, р.;

*Н -* накладные расходы, р.

Затраты на вспомогательные материалы приведены в таблице 20.

Таблица 20 - Затраты на вспомогательные материалы

|  |  |  |
| --- | --- | --- |
| Наименование затрат | Количество, шт. | Сумма, р. |
| 1. Тонер для заправки картриджей HP CF218A (№18A), HP CF230A (№18A) Тип 6.2, Bk, 50 г, банка, с воронкой-дозатором | 1 | 323 |
| 2. Бумага для офисной техники Снегурочка, А4, 80 г/кв. м, белизна 146% CIE, 500 листов | 1 | 379 |
| 3. Флешка Kingston DataTraveler 100 G316 ГБ | 1 | 449 |
| Итого: | | 1151 |

Затраты на электроэнергию рассчитываются по формуле:

,

Где

- мощность потребляемой электроэнергии, кВт;

** - стоимость одного киловатт-часа электроэнергии, р./кВт-ч;

* -* общие затраты труда на разработку программного продукта, час;

* -* коэффициент загрузки компьютера.

Подставив в формулу числовые значения вычислим затраты на электроэнергию:

*Э*=0,73·4,28·150·0,5=234, 33 *р.*

Основная заработная плата разработчика рассчитывается по формуле:

*Зозп=Счтс·Т´ОБЩ,*

где

 *-* часовая тарифная ставка разработчика, р./час;

- общие затраты труда на разработку программного продукта, час.

Подставив в формулу числовые значения вычислим основную заработную плату разработчика:

*Зозп=500\*150=75000* *р.*

Накладные расходы рассчитываются по формуле:

*Н=0,1·( Зозп+ Здзп )*,

Подставив числовые значения в формулу, получим:

*Н=* 0,1·(75000+0)=7500 *р.*

Полная себестоимость разработанного программного продукта:

*С*=1151+234, 33+75000+0+0+7500= 83885,33 *р.*

В таблице 21 сведены результаты расчетов себестоимости программного продукта.

Таблица 21 - Калькуляция себестоимости программного продукта

|  |  |
| --- | --- |
| Наименование статей расходов | Затраты, р. |
| 1 | 2 |
| 1. Основная заработная плата разработчика | 75000 |
| 2. Затраты на электроэнергию | 234, 33 |
| 3 Накладные расходы | 7500 |
| 4. Вспомогательные материалы | 1151 |
| 5. Полная себестоимость программного  продукта | 83885,33 |

Вывод: Себестоимость программного продукта составила 83885 рублей 33 копейки.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта на основе предпроектного исследования была выявлена актуальность работы, а также особенности создания систем данного вида, основанные на изучении сайтов конкурентов. Опираясь на результаты этих исследований, была построена структура системы и впоследствии разработана информационная система управления учебным временем студента.

Использование выбранных технических решений, важнейшими из которых являются язык программирования Php и фреймворк Yii, помогло достичь всех поставленных целей. Опираясь на приобретенные в процессе обучения знания, мне удалось воплотить весь намеченный функционал системы и совместить простоту и удобство ее интерфейса.

Разработанная информационная система имеет следующие преимущества:

* учитывает все нюансы и полностью соответствует существующей в организации системе ведения, контроля и оценки воспитательной работы;
* не требует инсталляции как самой информационной системы, так и дополнительных программных продуктов, библиотек и драйверов;
* не зависит от программного обеспечения рабочих мест сотрудников и функционирует в полном объеме на любом компьютере с любой версией ОС Windows;
* не требует никаких изменений, корректировок и доработок при смене местоположения информационной системы на базовом компьютере;
* обеспечивает многопользовательский режим работы аналогично архитектуре «клиент-сервер» без использования отдельных клиентских приложений;
* обеспечивает защиту персональных данных всех пользователей системы;
* имеет низкую себестоимость, благодаря удачно найденным решениям, значительно сократившим объем программного кода и время разработки.

Разработанная информационная система может служить основой для дальнейшей автоматизации информационных процессов при организации учебного времени. В дополнение к этому, внедрение системы позволит изменить существующий порядок взаимодействия лиц, организующих и проводящих воспитательную работу, в сторону значительного повышения ее эффективности и оперативности.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**Электронные ресурсы:**

1. Официальный сайт справочной документации по языку Php. - URL: <https://www.php.net/> (дата обращения: 20.04.2023).
2. Официальный сайт документации фреймворка Yii. - URL: <https://www.yiiframework.com/> (дата обращения: 20.04.2023).
3. Официальный сайт документации MySQL базы данных. - URL: <https://dev.mysql.com/> (дата обращения: 06.05.2023).
4. Официальный сайт документации Bootstrap. - URL: <https://bootstrap-4.ru/> (дата обращения: 13.05.2023).
5. Официальный сайт документации Visual Studio Code. - URL: <https://code.visualstudio.com/> (дата обращения: 23.04.2023).
6. Официальный сайт документации Open Server Panel. - URL: <https://ospanel.io/> (дата обращения: 21.04.2023).
7. Методы авторизации Yii. Настройка RBAC Manager. - URL: <https://www.yiiframework.com/doc/guide/2.0/ru/security-authorization> (дата обращения: 15.04.2023).
8. Расширения для Yii Framework 2.0. Виджет highcharts. - URL: <https://www.yiiframework.com/extension/yii2-highcharts-widget> (дата обращения: 18.05.2023).
9. Официальный сайт документации Krita 5.0. - URL: <https://docs.krita.org/en/index.html> (дата обращения: 17.05.2023).
10. James R. Bailey and Scheherazade Rehman Don’t Underestimate the Power of Self-Reflection: статья научно-популярного журнала Harvard Business Review. - URL: <https://hbr.org/2022/03/dont-underestimate-the-power-of-self-reflection?autocomplete=true> (дата обращения: 24.05.2023).
11. Бесплатный шаблон Bootstrap «Black Dashboard». - URL: <https://bootstraptema.ru/stuff/templates_bootstrap/admin/black_dashboard/4-1-0-4953> (дата обращения: 09.04.2023).
12. Pure CSS Radio Button Tiles - How to Design Neon Radio Buttons using HTML and CSS: открытый код для стилизации переключателей. - URL: <https://github.com/JamshidElmi/Pure-CSS-Radio-Button-Tiles> (дата обращения: 03.05.2023).
13. Расширения для Yii Framework 2.0. - URL: https://demos.krajee.com/ (дата обращения: 22.05.2023).

## ПРИЛОЖЕНИЕ А (обязательное) Программные коды

Метод в TaskController, реализующий вывод и выполнение задачи.

public function actionIndex()

{

$searchModel = new TaskSearch();

$dataProvider = $searchModel->search($this->request->queryParams);

$status = Status::getStatusList();

$subject = Subject::getSubjectList();

$priority = Priority::getPriorityList();

$mood = Mood::getMoodList();

$model = null;

$reflection = new Reflection();

if(Yii::$app->user->can('can\_admin')) {

$count = Task::find()->count();

}

if(Yii::$app->user->can('per\_user')) {

$count = Task::find()->where(['user\_id' => Yii::$app->user->id])->count();

if (Yii::$app->user->can('per\_student') && StudOfGroup::findOne(['student\_id' => Yii::$app->user->id])) {

$count += Task::find()->where(['group\_id' => StudOfGroup::findOne(['student\_id' => Yii::$app->user->id])->group\_id])->count();

}

}

if(Yii::$app->request->isPost)

{

$id = Yii::$app->request->get('id');

$check = Yii::$app->request->get('check');

if ($model = $this->findModel($id)) {

switch($check)

{

case 'true' :

$model->status\_id = Status::findOne(['title' => 'Выполнено'])->id;

$model->checked = 1;

$model->save();

break;

case 'false' :

$model->status\_id = Status::findOne(['title' => 'Не выполнено'])->id;

$model->checked = 0;

$model->save();

break;

}

}

if ($reflection->load($this->request->post())) {

if ($exist = Reflection::findOne(['task\_id' => $id])) {

$exist->delete();

}

$reflection->task\_id = $id;

$reflection->user\_id = Yii::$app->user->id;

$reflection->save();

return $this->redirect('index');

}

if ($model->status->title == 'Не выполнено' && ($ref = Reflection::findOne(['task\_id' => $id]))){

$ref->delete();

}

}

$not\_completed\_tasks = Task::find()->where(['status\_id' => Status::findOne(['title' => 'Не выполнено'])->id, 'group\_id' => null])->all();

foreach ($not\_completed\_tasks as $task)

{

if ($task->deadline\_date){

if (($task->deadline\_date . ' ' . $task->deadline\_time) <= date('Y-m-d H:i:s')){

$task->status\_id = Status::findOne(['title' => 'Просрочено'])->id;

$task->save();

}

}

}

return $this->render('index', [

'searchModel' => $searchModel,

'dataProvider' => $dataProvider,

'status' => $status,

'subject' => $subject,

'priority' => $priority,

'model' => $model,

'count' => $count,

'reflection' => $reflection,

'mood' => $mood,

]);

}

Метод в TaskController, реализующий редактирование задачи.

public function actionUpdate($id)

{

$model = $this->findModel($id);

$group = Group::getGroupList();

$subject = Subject::getSubjectList();

$model->groupName ? $prompt = $model->groupName->title : $prompt = '';

if ($this->request->isPost) {

$id = Yii::$app->request->get('id');

$file\_id = Yii::$app->request->get('file');

if ($file\_id) {

$delete = File::findOne($file\_id);

unlink(Yii::getAlias('@app') . '/web' . $delete->name);

$delete->delete();

return $this->redirect(['update', 'id' => $id]);

}

if ($model->load($this->request->post())) {

$model->add\_files = UploadedFile::getInstances($model, 'add\_files');

if ($model->add\_files){

if (count($model->add\_files) + count(TaskFiles::find()->where(['task\_id' => $model->id])->all()) <= 5){

if ($model->upload()) {

foreach (UploadedFile::getInstances($model, 'add\_files') as $file) {

$filename = Yii::$app->user->id . '\_' . date('d.m.y.H.i.s') . '\_' .$file->name;

$taskfiles = new TaskFiles();

$taskfiles->task\_id = $model->id;

$taskfiles->file\_id = File::find()->where(['name' => '/files/' . $filename])->one()->id;

$taskfiles->save();

}

} else {

Yii::$app->session->setFlash('error', 'Ошибка при редактировании!');

return $this->redirect(['update', 'id' => $id]);

}

} else {

return $this->redirect(['update', 'id' => $id]);

}

}

if ($model->group\_id != null || $model->important){

$model->priority\_id = Priority::getIdPriorityName('Важная');

} else {

if (!$model->important) {

$model->priority\_id = Priority::getIdPriorityName('Повседневная');

}

}

if ($model->deadline\_date && !$model->deadline\_time){

$model->deadline\_time = '23:59';

} elseif ($model->deadline\_time && !$model->deadline\_date){

$model->deadline\_date = date('Y-m-d');

}

if ($model->save(false)){

$model->group\_id = $model->group;

return $this->redirect(['view', 'id' => $model->id]);

}

}

}

return $this->render('update', [

'model' => $model,

'group' => $group,

'subject' => $subject,

'prompt' => $prompt,

'alert' => $alert,

]);

}