МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
  
 к курсовой работе по дисциплине «Программная инженерия»

по теме «Обучающая система по основам алгоритмизации

и программированию»

Обучающийся М.А. Мананников

Обучающийся Р.С. Рыбаков

Обучающийся И.Г. Горбачев

Обучающийся А.Д. Деревяшкин

Руководитель Л.С. Зеленко

Самара 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

ЗАДАНИЕ

на курсовую работу по дисциплине

«Программная инженерия»

обучающимся в группе № 6403-020302D

М.А. Мананников

Р.С. Рыбаков

И.Г. Горбачев

А.Д. Деревяшкин

1. Тема проекта:«Обучающая система по основам программирования и алгоритмизации»
2. Исходные данные к проекту**:** см. приложение к заданию
3. Перечень вопросов, подлежащих разработке:
   1. Произвести анализ предметной области: изучить методику составления учебных заданий и принципов разработки учебных программ
   2. Выполнить обзор существующих систем-аналогов
   3. Разработать информационно-логический проект системы по методологии UML
   4. Разработать и реализовать программное и информационное обеспечение, провести его тестирование и отладку
   5. Оформить документацию курсовой работы
   6. Подготовить презентацию по разработанной системе
4. Перечень графических разработок:
   1. Структурная схема системы
   2. Канонические диаграммы UML
   3. Схемы основных алгоритмов
5. Календарный план выполнения работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Содержание работы по этапам | Объем этапа в % к общему объему проекта | Срок  окончания | Фактическое выполнение |
| 1 | Оформление технического задания и его утверждение | 5 | 20.09.2024 |  |
| 2 | Описание и анализ предметной области | 10 | 27.09.2024 |  |
| 3 | Проектирование системы | 40 | 13.12.2024 |  |
| 3.1 | Разработка структурной схемы системы | 5 | 11.10.2024 |  |
| 3.2 | Разработка функциональной спецификации системы и прототипа интерфейса пользователя | 10 | 25.10.2024 |  |
| 3.3 | Разработка информационно-логического проекта системы и его предъявление руководителю | 25 | 13.12.2024 |  |
| 4 | Реализация проекта, разработка контрольных примеров. Предъявление реализации руководителю | 40 | 13.12.2024 |  |
| 5 | Корректировка проекта и оформление документации проекта. Защита проекта с представлением презентации. | 5 | 27.12.2024 |  |

Задание принял  
к исполнению М.М. Мананников

Р.С. Рыбаков

И.Г. Горбачев

А.Д. Деревяшкин

ПРИЛОЖЕНИЕ  
к заданию на курсовую работу  
обучающимся в группе № 6403-020302D  
М.А. Мананников

Р.С. Рыбаков

И.Г. Горбачев

А.Д. Деревяшкин

Тема проекта: «Обучающая система по основам алгоритмизации и программированию»

Исходные данные к проекту:

1. Характеристика объекта автоматизации:

## объект автоматизации: обучающая система;

## виды автоматизируемой деятельности:

* + процесс авторизации пользователей;
  + процесс генерирования шаблона игрового поля;
  + процесс визуализации игрового поля;
  + процесс автоматической расстановки препятствий и персонажа;
  + процесс выполнения учебного задания;
  + процесс визуализации выполнения задания;

## количество ролей пользователей – 2;

## минимальная длина логина – 3 символа;

## максимальная длина логина – 15 символов;

## минимальная длина пароля – 8 символа;

## максимальная длина пароля – 15 символов;

## минимальный размер по горизонтали – 4 клетки;

1. максимальный размер по горизонтали – 10 клеток;
2. минимальный размер по вертикали – 4 клетки;
3. максимальный размер по вертикали – 10 клеток;
4. количество способов создания игрового поля – 2;
5. количество шаблонов игрового поля – 5;
6. количество персонажей – 1;
7. минимальное количество кубиков – 0;
8. максимальное количество кубиков – 15% от размера игрового поля;
9. минимальное количество лунок – 0;
10. максимальное количество лунок – 10% от размера игрового поля;
11. количество уровней сложности – 3;
12. максимальное количество вложенных циклов – 3;
13. максимальное количество итераций цикла – 10;
14. минимальная длина текста задания – 10 символов;
15. максимальная длина текста задания – 100 символов;
16. количество шаблонов команд – 11;
17. максимальное количество команд в программе – 100;
18. максимальное количество учеников – 100;
19. система оценивания – двухбалльная;
20. максимальное количество текстов задания – 50;
21. максимальное количество игровых полей – 50.
22. Требования к информационному обеспечению:
23. информационное обеспечение разрабатывается на основе следующих источников:
    * система программирования Кумир [Электронный ресурс]. URL: https://www.niisi.ru/kumir/ (дата обращения: 14.09.2024);
24. персонаж и препятствия располагаются произвольно на игровом поле;
25. расположение персонажа и препятствия не должны совпадать;
26. в каждом игровом поле необходимо наличие как минимум 1 алгоритма решения задания;
27. структура базы данных (БД) разрабатывается на основании следующих сведений:
    * об ученике (ФИО, логин, пароль);
    * об игровом поле (название игрового поля, структура игрового поля, уровень сложности);
    * об уровне сложности (размеры по горизонтали и вертикали, количество кубиков, лунок и занятых клеток);
    * о журнале (ученик, список оценок);
28. в системе должен вестись справочник текстов к заданиям;
29. должна быть обеспечена целостность базы данных и защита от несанкционированного доступа.
30. Требования к техническому обеспечению:
    1. Требования к техническому обеспечению серверной части:
31. тип ЭВМ – IBM PC совместимый;
32. объем ОЗУ – не менее 1 Гб;
33. объем свободного пространства на внешнем диске – не менее 20 Гб;
34. наличие подключения к сети Интернет;
35. манипулятор – мышь;
36. устройство ввода – клавиатура;
37. технические характеристики определяются в процессе выполнения проекта.
    1. Требования к техническому обеспечению клиентской части:
38. тип ЭВМ – IBM PC совместимый;
39. монитор с разрешающей способностью не ниже 800 х 600;
40. манипулятор – мышь;
41. устройство ввода – клавиатура;
42. технические характеристики определяются в процессе выполнения проекта.
43. Требования к программному обеспечению:
    1. Требования к программному обеспечению серверной части:
44. тип операционной системы – Windows 7 и выше;
    1. Требования к программному обеспечению клиентской части:
45. тип операционной системы – Windows 7 и выше;
46. браузер – Google Chrome;
    1. Требования к программному обеспечению рабочего места разработчика:
47. тип операционной системы – Windows 7 и выше;
48. язык программирования – Python 3.12;
49. фреймворк – Django 5.1.2;
50. среда программирования – Visual Studio Code;
51. среда проектирования – StarUML 5.2.0.
52. Общие требования к проектируемой системе:

5.1 Функции, реализуемые системой:

1. функции системы:
   * аутентификация пользователя в системе, настройка интерфейса пользователя на заданную роль;
   * работа с базой данных (БД):
   * сохранение данных;
   * загрузка данных;
   * контроль целостности;
   * формирование списка заданий;
   * контроль составленной программы;
   * визуализация работы с игровым полем;
   * визуализация выполнения программы;
   * выдача справочной информации о системе;
2. функции учителя:
   * авторизация пользователя в системе (ввод логина и пароля);
   * работа с учеником:
   * блокировка ученика;
   * работа с журналом:
   * выбор ученика;
   * выбор задания;
   * выставление оценок;
   * просмотр журнала;
   * настройка уровня сложности:
     + задание размеров по горизонтали;
     + задание размеров по вертикали;
     + задание количества кубиков;
     + задание количества лунок;
     + задание количества занятых клеток;
     + выбор уровня сложности;
     + создание игрового поля;
     + выбор способа создания игрового поля;
     + ручное создание игрового поля:
     + выбор шаблона;
     + определение местоположения шаблона;
     + сохранение игрового поля в БД;
     + создание текста задания;
     + формирование задания:
     + выбор игрового поля;
     + выбор текста задания;
     + загрузка игрового поля из БД;
     + просмотр игрового поля;
     + просмотр справочной информации;
     + выполнение задания ученика;
3. функции ученика:
   * авторизация пользователя в системе (ввод логина и пароля);
   * отправка команд на проверку учителю;
   * выбор уровня сложности;
   * выбор задания;
   * составление программы:
   * выбор шаблона команды;
   * настройка параметров цикла;
   * определение расположения команды в программе;
   * изменение команды;
   * запуск на выполнение;
   * сохранение в БД программы;
   * отправка решения учителю;
   * просмотр собственной успеваемости;
   * просмотр справочной информации.

5.2 Технические требования к системе:

1. режим работы – диалоговый;
2. время автоматической генерации игрового поля – не более 10 с;
3. система должна удовлетворять санитарным правилам и нормам  
    СанПин 2.2.2./2.4.2198-07;
4. условия работы средств вычислительной техники (содержание вредных веществ, пыли и подвижность воздуха) должны соответствовать ГОСТ 12.1.005, 12.01.007;
5. температура окружающего воздуха – 15-35°С;
6. влажность воздуха – 45-75%.

Руководитель   
проекта Л.С. Зеленко

Задание принял  
к исполнению М.М. Мананников

Р.С. Рыбаков

И.Г. Горбачев

А.Д. ДеревяшкинРЕФЕРАТ

Пояснительная записка 117 с, 52 рисунка, 16 таблиц, 29 источников,  
2 приложения.

Графическая часть: 24 слайдов презентации PowerPoint.

ИГРОВОЕ ПОЛЕ, ОБУЧАЮЩАЯ ПЛАТФОРМА, УРОВЕНЬ СЛОЖНОСТИ, ГЕНЕРАЦИЯ ИГРОВОГО ПОЛЯ, АЛГОРИТМ СОЗДАНИЯ, АЛГОРИТМ РЕШЕНИЯ, ПЕРСОНАЖ.

Объектом автоматизации является игровое поле.

Во время курсового проектирования разработаны алгоритмы и соответствующая им программа, позволяющая выполнять автоматическую и ручное создание игрового поля по заданным параметрам. В системе предусмотрены две роли: учитель и ученик. Учитель является администратором в системе и имеет расширенный функционал. Он может проставлять оценки в журнале, создавать задания для учеников и блокировать их в системе. Ученик может редактировать личный профиль, просматривать свою успеваемость и выполнять задания путем написания кода программы.

Программа написана на языке Python в среде Visual Studio Code и функционирует под управлением операционной системы Windows 7 и выше. Доступ к данным осуществляется с помощью файлов.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 13](#_Toc186202277)

[1 Описание и анализ предметной области 15](#_Toc186202278)

[1.1 Описание предметной области 15](#_Toc186202279)

[1.1.1 Типы обучающих систем 16](#_Toc186202280)

[1.2 Описание систем-аналогов 17](#_Toc186202281)

[1.1.1 Обучающая платформа «Кумир» 17](#_Toc186202282)

[1.2.2 Обучающая платформа «Scratch» 19](#_Toc186202283)

[1.3 Диаграмма объектов предметной области 23](#_Toc186202284)

[1.4 Постановка задачи 24](#_Toc186202285)

[1.1.2 Режим учителя 24](#_Toc186202286)

[1.4.3 Режим ученика 25](#_Toc186202287)

[2 Проектирование системы 29](#_Toc186202288)

[2.1 Выбор и обоснование архитектуры системы 29](#_Toc186202289)

[2.2 Структурная схема системы 33](#_Toc186202290)

[2.3 Разработка спецификации требований 35](#_Toc186202291)

[2.3.1 Функциональная спецификация 37](#_Toc186202292)

[2.3.2 Перечень исключительных ситуаций 37](#_Toc186202293)

[2.4 Разработка прототипа интерфейса пользователя системы 41](#_Toc186202295)

[2.5 Разработка информационно-логического проекта системы 48](#_Toc186202296)

[2.5.3 Диаграмма вариантов использования 51](#_Toc186202297)

[2.5.4 Сценарии 56](#_Toc186202298)

[2.5.5 Диаграмма классов 64](#_Toc186202308)

[2.5.6 Диаграмма состояний 66](#_Toc186202309)

[2.5.7 Диаграмма деятельности 70](#_Toc186202310)

[2.5.8 Диаграмма последовательности 71](#_Toc186202311)

[2.6 Логическая модель данных 76](#_Toc186202312)

[2.7 Выбор и обоснование комплекса программных средств 79](#_Toc186202313)

[2.7.1 Выбор языка программирования 80](#_Toc186202314)

[2.7.2 Выбор фреймворков 81](#_Toc186202315)

[2.7.3 Выбор среды программирования 82](#_Toc186202316)

[2.7.4 Выбор операционной системы 83](#_Toc186202317)

[2.7.5 Выбор системы управления базами данных 84](#_Toc186202318)

[3 Реализация системы 85](#_Toc186202319)

[3.1 Разработка и описание интерфейса пользователя 85](#_Toc186202320)

[3.2 Диаграммы реализации 92](#_Toc186202321)

[3.2.1 Диаграмма компонентов 92](#_Toc186202322)

[3.2.2 Диаграмма развертывания 95](#_Toc186202323)

[3.2.3 Диаграмма классов 96](#_Toc186202324)

[3.2.4 Физическая модель базы данных 96](#_Toc186202325)

[3.3 Выбор и обоснование комплекса технических средств 99](#_Toc186202326)

[3.3.1 Расчет объема занимаемой памяти 100](#_Toc186202327)

[3.3.2 Минимальные требования, предъявляемые к системе 102](#_Toc186202331)

[ЗАКЛЮЧЕНИЕ 104](#_Toc186202332)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 105](#_Toc186202333)

[ПРИЛОЖЕНИЕ А Руководство пользователя 108](#_Toc186202334)

[А.1 Назначение системы 108](#_Toc186202335)

[А.2 Условия работы системы 108](#_Toc186202336)

[А.3 Установка системы 109](#_Toc186202337)

[А.4 Работа с системой 109](#_Toc186202338)

[А.4.1 Работа с системой в режиме пользователя 111](#_Toc186202339)

[А.4.1 Работа с системой в режиме учителя 113](#_Toc186202340)

[ПРИЛОЖЕНИЕ Б Листинг модулей программы 115](#_Toc186202341)

ВВЕДЕНИЕ

В последние годы наблюдается растущий интерес к использованию игровых элементов в образовательных системах, особенно в таких сложных областях, как алгоритмизация и программирование. Это связано с тем, что традиционные методы обучения часто оказываются недостаточно эффективными для вовлечения обучающихся и формирования у них необходимых навыков.

Игровое поле может быть представлено как физическая или виртуальная среда, где учащиеся могут взаимодействовать с различными элементами, решать задачи и достигать поставленных целей. Оно создает контекст, в котором учебный процесс становится более динамичным и увлекательным. Игровые механики, такие как уровни, достижения и награды, могут значительно повысить мотивацию обучающихся к обучению, развивать критическое мышление и способствовать сотрудничеству.

Алгоритмизация – это основополагающий процесс в программировании, который включает в себя разработку четких последовательностей действий для решения задач. Игровое поле может быть использовано для визуализации этих процессов. Например, визуальные языки программирования, такие как Scratch или Blockly, позволяют учащимся создавать алгоритмы с помощью перетаскивания блоков, что делает процесс более интуитивным и доступным. Платформы вроде CodeCombat или LightBot предлагают учащимся решать задачи программирования в формате игры, что делает обучение более захватывающим.

Создание тематических учебных заданий помогает учащимся проходить через различные уровни сложности, решая алгоритмические задачи. Введение системы оценивания помогает отслеживать прогресс обучающегося, а также способствует повышению их вовлеченности.

Использование игрового поля в обучении алгоритмизации и программированию способствует развитию ряда ключевых навыков. Обучающиеся учатся разбивать сложные задачи на более мелкие подзадачи и находить оптимальные решения, что развивает их логическое мышление. Игровой формат позволяет учащимся экспериментировать с различными подходами к решению задач, что способствует развитию творческого мышления. Кроме того, игровые ситуации часто включают ошибки и неудачи, что помогает учащимся научиться справляться с трудностями и искать альтернативные пути решения.

Во время курсового проектирования необходимо разработать обучающую систему по основам программирования и алгоритмизации, с помощью которой учитель сможет создать задание различной сложности, а ученик с помощью различных алгоритмов составить программу для решения задания.

Разработка системы будет производиться по технологии быстрой разработки приложений RAD (Rapid Application Development)*,* которая поддерживается методологией структурного проектирования и включает элементы объектно-ориентированного проектирования и анализа предметной области [3].

При проектировании системы будет использоваться методология ООАП (Object-Oriented Analysis/Design), в основу которой положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов. Также будут использоваться методология MVT (Model View Template) – это архитектурный паттерн, который используется в веб-разработке, особенно в рамках фреймворка Django и язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения [4].

1. Описание и анализ предметной области

Под предметной областью (application domain) принято понимать ту часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Другими словами, предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи [1].

* 1. Описание предметной области

Обучающая система – это комплекс программных и технических средств, предназначенных для организации, управления и сопровождения учебного процесса. Она помогает автоматизировать процесс обучения, обеспечивая интерактивное взаимодействие учащихся с учебными материалами, преподавателями и между собой, а также контролируя прогресс и результаты обучения [2].

Игровое поле – это пространство или среда, в которой разворачивается игровая активность, включая правила, объекты и взаимодействия между игроками. Оно может быть как физическим (например, настольные игры), так и виртуальным (например, видеоигры), и служит основой для игрового процесса.

Процесс обучения – это целенаправленная передача знаний и навыков от учителя к ученикам с целью их развития и закрепления. Учитель планирует учебный процесс, подает материал, стимулирует активное участие и самостоятельное мышление учеников. Важную роль играет обратная связь, которая помогает оценить ученика и корректировать обучение. Учитель также поддерживает мотивацию учеников, организует групповую работу и способствует развитию навыков самостоятельного обучения. Взаимодействие учителя и учеников строится на открытой коммуникации, создавая условия для эффективного обучения и личностного роста [3].

Журнал – это документ или электронная система, используемая для фиксации и отслеживания событий, результатов или действий, связанных с учебным процессом [4]. В учебных системах журнал помогает регистрировать посещаемость, оценки, выполненные задания и обратную связь, создавая полноценную картину успеваемости и активности учащихся. Учителя и администраторы могут использовать журнал для анализа прогресса учеников и выявления областей, требующих дополнительного внимания.

В обучающей системе ведётся электронный журнал, где отмечаются посещения студентов на занятиях, полученные оценки за тесты и домашние задания, а также комментарии преподавателя. Это позволяет как учителю, так и ученику отслеживать прогресс и видеть общую картину обучения.

* + 1. Типы обучающих систем

Обучающие системы можно классифицировать на несколько типов в зависимости от их структуры и целей [5]:

1. локальные обучающие системы – это программы, которые устанавливаются на компьютер или работают в локальной сети. Они используются для индивидуального обучения и обычно не требуют подключения к интернету. Примеры включают тренажеры и обучающие игры;
2. дистанционные обучающие системы – позволяют обучаться удаленно через интернет. Эти системы обеспечивают доступ к курсам, тестам и заданиям онлайн, а также взаимодействие с преподавателями и другими учащимися. Примеры: платформы «Moodle», «Coursera», «Google Classroom»;
3. интерактивные обучающие системы – включают в себя элементы взаимодействия с пользователем, предлагая задания, выборы, тесты и обратную связь в реальном времени. Такие системы помогают учащимся активно применять полученные знания на практике;
4. интеллектуальные обучающие системы – используют технологии искусственного интеллекта для адаптации процесса обучения под потребности ученика. Они анализируют успеваемость и подстраивают учебные материалы для оптимального развития навыков;
5. гибридные обучающие системы – объединяют элементы традиционного (очного) и дистанционного обучения, предлагая часть материала в классе, а часть – через онлайн-платформы. Такой подход часто применяется в смешанном обучении.

Эти типы обучающих систем помогают автоматизировать и оптимизировать процесс обучения, делая его доступным, гибким и персонализированным.

* 1. Описание систем-аналогов

В интернете существует несколько вариантов обучающих платформ. Все они разные и имеют свои особенности и функциональные свойства. Разберём несколько систем-аналогов проектируемой системе и рассмотрим их достоинства и недостатки.

* + 1. Обучающая платформа «Кумир»

«КуМи́р» (Комплект Учебных МИРов или Миры Кушниренко) – язык и система программирования, предназначенная для поддержки начальных курсов информатики и программирования в средней и высшей школе [6]. В 1995 году «КуМир» был рекомендован Министерством образования РФ в качестве основного учебного материала по курсу «Основы информатики и вычислительной техники» на основе учебника А.Г. Кушниренко, Г.В. Лебедева и Р.А. Свореня. Обучающая система основана на методике, разработанной во второй половине 1980-х годов под руководством академика А.П. Ершова. Эта методика широко использовалась в средних школах СССР и России. В системе «КуМир» используется придуманный А. П. Ершовым школьный алгоритмический язык – простой алголоподобный язык с русской лексикой и встроенными командами управления программными исполнителями (Робот, Чертёжник).

В настоящее время разработана и развивается новая версия «КуМир», использующая библиотеку «Qt» и работающая в операционных системах Linux и Windows. Постановка задачи на разработку новой версии была выполнена А.Г. Кушниренко и А.Г. Леоновым. Разработка ведётся пущинской группой сотрудников ФГУ ФНЦ НИИСИ под руководством М.А. Ройтберга.

«КуМир» использует блочную среду, где блоки с командами перетаскиваются и соединяются в последовательность. Также «КуМир» предлагает опцию для написания кода на текстовом языке Паскаль, что позволяет постепенно переходить от визуального программирования к текстовому. Визуализация алгоритмов – еще одна важная функция «КуМир», которая помогает понять логику и действие алгоритмов. Система предлагает широкий спектр задач, включая задачи с графикой, текстом, числами и другими элементами.

Система «КуМир» используется в российских школах для обучения алгоритмизации и программированию. В рамках ОГЭ и ЕГЭ по информатике ученики выполняют задания с исполнителями, такими как Робот, что помогает проверять их умение строить алгоритмы. Для сдачи экзаменов часто требуется написать программу на языке «КуМир» или создать пошаговый алгоритм решения задачи, используя исполнителей.

На рисунке 1 изображено игровое поле обучающей платформы «КуМир».

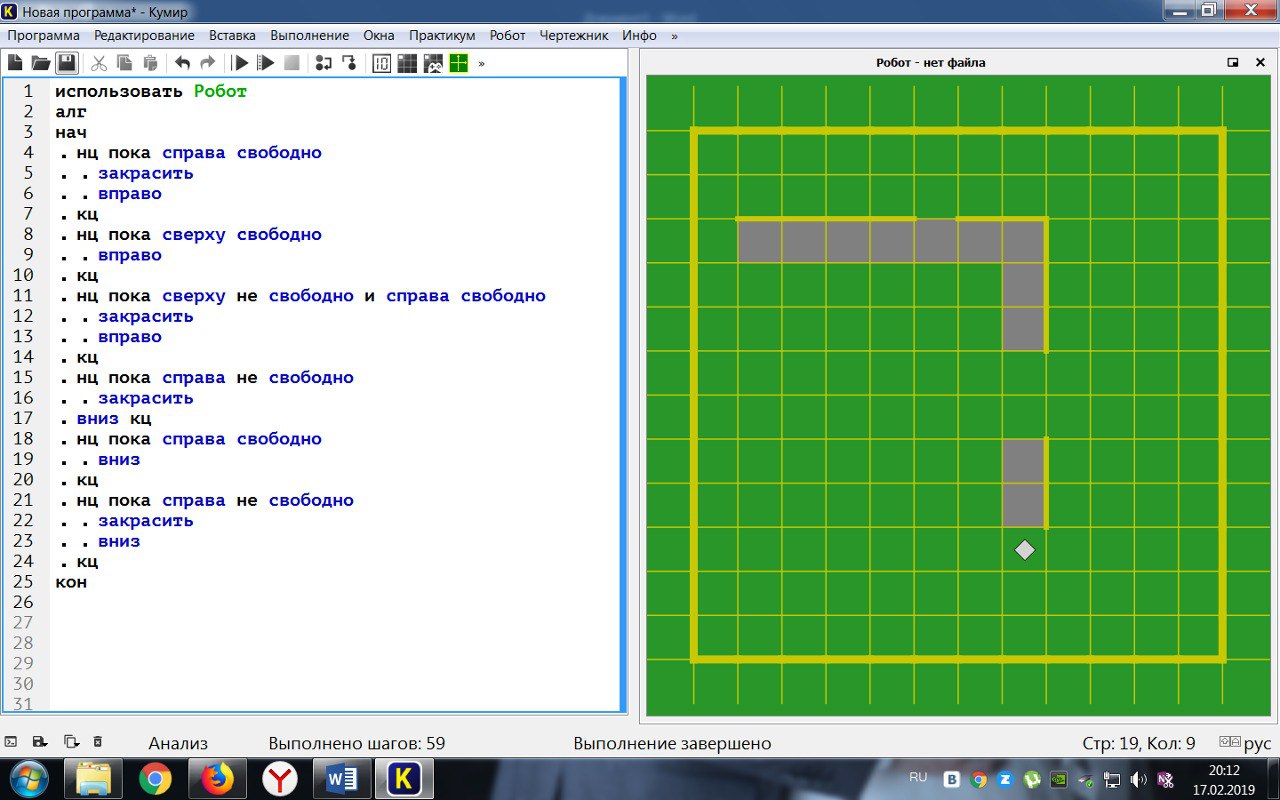


Рисунок 1 – Игровое поле обучающей платформы «КуМир»

К достоинствам данной системы относятся:

* визуальное программирование блоками облегчает освоение основ алгоритмизации, что делает среду понятной для новичков;
* текстовый язык позволяет плавно переходить к написанию кода на реальном языке программирования;
* визуализация помогает в понимании логики алгоритмов и процесса выполнения.

К недостаткам системы относятся:

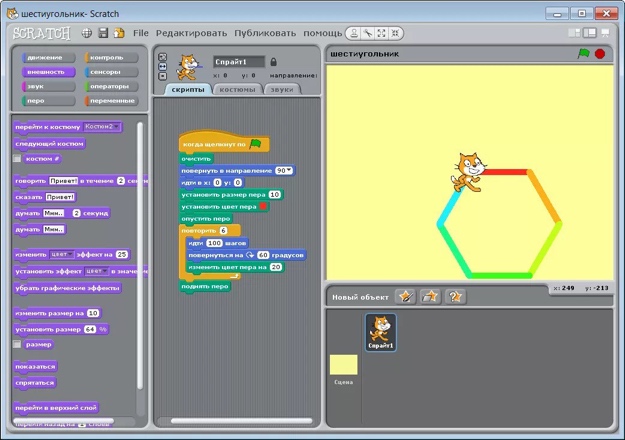
* среда программирования выглядит устаревшей, что может отпугнуть пользователей;
* возможности Кумира в сравнении с другими системами ограничены;
* отсутствие мобильной версии делает его менее удобным для использования на мобильных устройствах.
  + 1. Обучающая платформа «Scratch»

Язык программирования «Scratch» был разработан в Массачусетском технологическом институте, в «MIT Media Lab» в 2007 году. Целью исследовательской группы было создание платформы для юных программистов. «Scratch» основан на принципе перетаскивания ярких визуальных блоков. Он состоит из наглядных инструментов для разработки игр, анимации, интерактивных историй. Также в процессе создания собственных проектов дети обучаются работе с графикой и звуком. Название данного языка программирования переводится как «царапать». Лучше его суть отражает фраза «From scratch», которая переводится «С самого начала» [7].

Каждый месяц платформу посещают более 38 миллионов пользователей, что является отличным показателем для онлайн-ресурса. Пользователи со всего мира делятся своими проектами, играми, анимацией, интерактивными историями. Scratch переведен на 17 языков, в том числе русский. Кроме того, он включен в программу различных тематических школ, летних лагерей и специализированных курсов программирования для начинающих.

«Scratch» использует блоки с командами для построения сценариев. Это делает процесс программирования доступным и понятным для пользователей. Кроме того, «Scratch» поддерживает объектно-ориентированное программирование, позволяя создавать объекты с собственными свойствами и поведением, что способствует лучшему пониманию основ программирования. Визуальная среда «Scratch» предоставляет интуитивно понятный графический интерфейс, который облегчает обучение.

На рисунке 2 изображено игровое поле обучающей платформы «Scratch».

  
Рисунок 2 – Игровое поле обучающей платформы Scratch

К достоинствам данной системы относятся:

* можно выделить его простоту в использовании, что делает его удобным к начинающим программистам. В
* визуализация процессов с помощью графических блоков и персонажей делает программирование более интерактивным и увлекательным;
* позволяет вводить базовые концепции объектно-ориентированного программирования с раннего возраста, что помогает детям лучше усваивать материал.

К недостаткам системы относятся:

* в некоторых случаях он может быть ограничен по функциональности для сложных проектов, что может разочаровать более опытных пользователей;
* отсутствие текстового языка программирования может затруднить переход к изучению более сложных языков, что ограничивает дальнейшее развитие навыков программирования.

На основании анализа возможностей систем-аналогов были сформулированы требования к разрабатываемой системе (см. таблицу 1).

Таблица 1 – Сравнительные характеристики систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название системы  Название показателя | «КуМир» | «Scratch» | Разрабатываемая система |
| Развитие навыков алгоритмизации | + | – | + |
| Поддержка русского языка | + | + | + |
| Кросс-платформенность | – | – | – |
| Современный пользовательский интерфейс | – | + | + |
| Написание кода программы | + | – | + |
| Визуализация выполнения алгоритма | + | – | + |
| Масштабируемость | – | – | + |

* 1. Диаграмма объектов предметной области

Обратная декомпозиция – это процесс анализа сложной системы или объекта, при котором он разбивается на более простые компоненты с целью понимания его структуры, функций или работы. Этот метод часто используется в инженерии, программировании и научных исследованиях для выявления и изучения взаимосвязей между частями системы [8].

Методология ООАП (Object-Oriented Analysis/Design) – это подход к проектированию систем, основанный на объектно-ориентированной методологии, который представляет предметную область в виде объектов, являющихся экземплярами соответствующих классов.

На рисунке 4 приведена диаграмма объектов предметной области. На диаграмме центральным объектом является игровое поле, которое представляет собой совокупность следующих объектов: клеток и персонажа, который передвигает кубик.

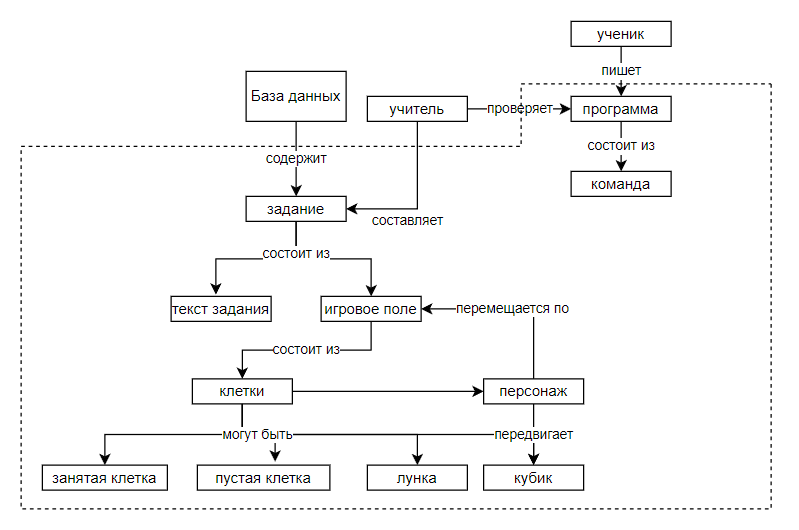


Рисунок 4 - Диаграмма объектов предметной области

Диаграмма объектов представляет статическую составляющую взаимодействующих между собой объектов, она должно включать в себя только те объекты предметной области, которые потом преобразуются в диаграмму классов. Связи между объектами показывают отношения между ними, при необходимости в диаграмме можно привести и атрибуты (свойства) объектов.

Диаграммы объектов не позволяют полностью описать объектную структуру системы, поэтому при их использовании нужно сосредоточиться на изображении интересующих вас наборов конкретных объектов.

* 1. Постановка задачи

Во время курсового проектирования необходимо разработать обучающую систему, предназначенную для ознакомления пользователей (далее – учеников) с основными концепциями алгоритмов и программирования. Система должна быть реализована в виде веб-приложения.

В системе должно быть реализовано две роли пользователей: учитель и ученик. Каждый пользователь при входе в систему должен выбрать одну из двух предложенных ролей, чтобы авторизоваться в системе: если учитель ввести пароль (от 8 до 15 символов); если ученик – логин (от 3 до 15 символов).

* + 1. Режим учителя

Основные задачи учителя – настройка уровня сложности, составление учебного задания, а также последующая проверка алгоритма решения задания учеником и выставление оценки по двухбалльной шкале: зачёт или незачёт.

Создание игрового поля будет проводиться в соответствии с текстом задания (длина от 10 до 100 символов), который составляет учитель. Учитель будет задавать параметры уровня сложности вручную – устанавливать размеры игрового поля по горизонтали (от 4 до 10 клеток) и вертикали (от 4 до 10 клеток), количество кубиков (от 0 до 15% от размеров игрового поля) и количество лунок (от 0 до 10% от игрового поля). Также учитель должен выбрать способ создания игрового поля: вручную (выбор шаблона и его местоположения) или автоматически. При автоматической способе система должна сгенерировать игровое поле в соответствии с заданными параметрами.

Также учитель сможет осуществлять удаление игрового поля и сохранение его в БД, при этом система должна провести проверку правильности этой структуры и, в случае несоответствия, выдать предупредительное сообщение и обеспечить повторное сохранение записи в БД. В системе должен осуществляться контроль значений параметров для проверки правильности размера игрового поля и препятствий.

Структура базы данных разрабатывается на основании следующих сведений:

* + об ученике (ФИО, логин, пароль);
  + об игровом поле (название игрового поля, структура игрового поля, уровень сложности);
  + об уровне сложности (размеры по горизонтали и вертикали, количество кубиков, лунок и занятых клеток);
  + журнал успеваемости (ученик, список оценок).

В режиме учителя будет реализована работа с учеником – учитель сможет добавлять, редактировать и удалять сведения об ученике; удалять ученика из базы данных или блокировать ему доступ к системе. Также будет реализована работа с журналом, в котором можно выбрать ученика и поставить оценку.

* + 1. Режим ученика

Основная задача ученика – выполнение задания, включающего написание кода для управления персонажем с кубиком так, чтобы тот обходил препятствия и достигал цели. Ученик может либо использовать специальные кнопки для выбора действий персонажа, либо вручную вводить команды в виде программного кода. После написания кода ученик отправляет его на проверку учителю, и, если последовательность действий правильна, задание считается выполненным.

Система должна формировать список заданий, контролировать код на соответствие заданным параметрам и сообщать о несоответствиях. Также будет визуализироваться процесс взаимодействия с игровым полем и выполнения программы, позволяя ученику видеть результат своих действий в реальном времени.

Таким образом, система должна решать следующие задачи:

1. функции системы:
   * аутентификация пользователя в системе, настройка интерфейса пользователя на заданную роль;
   * работа с базой данных (БД):
   * сохранение данных;
   * загрузка данных;
   * контроль целостности;
   * формирование списка заданий;
   * контроль составленной программы;
   * визуализация работы с игровым полем;
   * визуализация выполнения программы;
   * выдача справочной информации о системе;
2. функции учителя:
   * авторизация пользователя в системе (ввод пароля);
   * работа с учеником:
   * блокировка ученика;
   * работа с журналом:
   * выбор ученика;
   * выбор задания;
   * выставление оценок;
   * просмотр журнала;
   * настройка уровня сложности:
     + задание размеров по горизонтали;
     + задание размеров по вертикали;
     + задание количества кубиков;
     + задание количества лунок;
     + задание количества занятых клеток;
     + создание игрового поля;
     + выбор способа создания игрового поля;
     + ручное создание игрового поля:
     + выбор шаблона;
     + определение местоположения шаблона;
     + сохранение игрового поля в БД;
     + создание текста задания;
     + формирование задания:
     + выбор игрового поля;
     + выбор текста задания;
     + загрузка игрового поля из БД;
     + просмотр игрового поля;
     + просмотр справочной информации;
     + выполнение задания ученика;
3. функции ученика:
   * авторизация пользователя в системе (ввод логина);
   * отправка команд на проверку учителю;
   * выбор уровня сложности;
   * выбор задания;
   * составление программы:
   * выбор шаблона команды;
   * настройка параметров цикла;
   * определение расположения команды в программе;
   * изменение команды;
   * запуск на выполнение;
   * сохранение в БД программы;
   * отправка решения учителю;
   * просмотр собственной успеваемости;
   * просмотр справочной информации.
4. Проектирование системы

Проектирование – это процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части.

На этапе проектирования системы нужно решить ряд задач:

* выбрать архитектуру системы и разработать структурную схему системы;
* разработать прототип пользовательского интерфейса;
* разработать структуру базы данных;
* выбрать программные средства разработки.
  1. Выбор и обоснование архитектуры системы

Архитектура приложения – это структурный и концептуальный фреймворк, определяющий организацию и взаимодействие компонентов программного приложения. Она описывает, как различные части приложения взаимодействуют между собой, как данные передаются и обрабатываются, а также определяет принципы, которыми руководствуется разработка приложения [9].

Рассмотрим основные виды архитектур и их особенности:

* монолитная архитектура – это организация компонентов приложения, взаимодействие между ними и организация данных внутри приложения. Компоненты взаимодействуют друг с другом, чтобы обеспечить функциональность приложения;
* микросервисная архитектура: особенностями микросервисной архитектуры являются ее гибкость в разработке и масштабируемость. Каждый сервис может быть разработан с использованием разных технологий и языков программирования, развернут и обновлен отдельно, без воздействия на другие компоненты;
* сервис-ориентированная архитектура: компоненты приложения взаимодействуют через общие сервисы, что упрощает интеграцию и повторное использование;
* клиент-серверная архитектура: приложение разделено на клиентскую и серверную части, что обеспечивает лучшую управляемость и возможность масштабирования.

Клиент-серверная архитектура разделяет систему на две основные части: клиент и сервер. Клиент запрашивает информацию или услуги у сервера, а сервер обрабатывает запросы и предоставляет данные. Это позволяет централизовать управление данными и улучшить безопасность. Сервер может обслуживать множество клиентов одновременно, что делает архитектуру масштабируемой. Эта архитектура хорошо подходит для сетевых приложений.

На рисунке 5 показана схема клиент-серверной архитектуры.

Виды клиент-серверных архитектур:

* двухзвенная архитектура. Состоит из клиента и сервера. Клиент отправляет запросы, а сервер отвечает на них. Эта архитектура проста в реализации, но может иметь ограничения в производительности при большом количестве пользователей. На рисунке 5 представлена схема двухзвенной архитектуры [10].

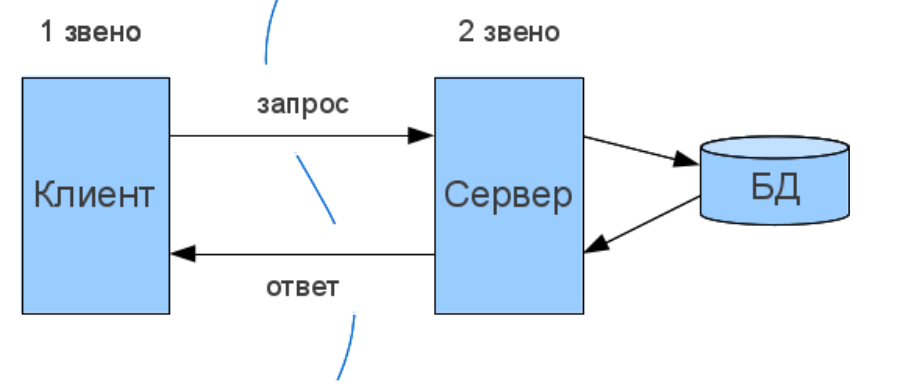


Рисунок 5 – Схема двухзвенной архитектуры

* трехзвенная архитектура. Включает клиент, сервер приложений и базу данных. Это позволяет разделить бизнес-логику и управление данными, что повышает масштабируемость и гибкость. На рисунке 6 представлена схема трехзвенной архитектуры [11].

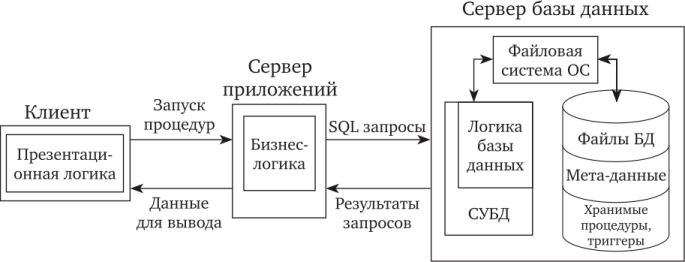


Рисунок 7 – Схема трехзвенной архитектуры

Для разрабатываемого приложения наиболее подходящим вариантом является клиент-серверная архитектура, потому что выбранная архитектура:

* обеспечивает разделение обязанностей: клиент-серверная архитектура позволяет отделить пользовательский интерфейс (клиент) от логики приложения и обработки данных (сервер). Это облегчает разработку, сопровождение и тестирование отдельных компонентов системы.
* масштабируема: в случае увеличения нагрузки на систему можно относительно легко масштабировать серверную часть, добавляя больше ресурсов или распределяя запросы между несколькими серверами, что позволяет поддерживать высокую производительность при больших объемах данных и пользователей.
* обеспечивает безопасность данных: сервер выступает центральной точкой обработки и хранения данных, что упрощает управление доступом, контроль безопасности и резервное копирование.
* поддерживает многопользовательский режим: клиент-серверная архитектура обеспечивает возможность одновременной работы множества пользователей.

Веб-приложение – это приложение, которое работает через веб-браузер и использует веб-технологии (HTML, CSS, JavaScript) для взаимодействия с пользователем. Веб-приложения могут работать на различных устройствах и не требуют установки на клиентском устройстве [12].

Протокол – это набор правил и стандартов, которые определяют, как данные передаются по сети. Он устанавливает формат сообщений и правила обмена между клиентом и сервером [13].

Рассмотрим основные типы протоколов:

* HTTP/HTTPS (Hyper Text Transfer Protocol/Security): протокол передачи гипертекстовой информации, используемый для загрузки веб-страниц. HTTPS обеспечивает безопасное соединение с использованием SSL/TLS;
* TCP/IP (Transmission Control Protocol/Internet Protocol): протокол IP отвечает за маршрутизацию пакетов данных между устройствами по IP-адресам, а TCP обеспечивает надежную передачу данных, гарантируя их целостность и порядок доставки;
* WebSocket: протокол для двустороннего взаимодействия между клиентом и сервером в реальном времени;
* FTP (File Transfer Protocol): протокол для передачи файлов между клиентом и сервером.

Для разрабатываемого приложения будет использоваться HTTP/HTTPS, так как он является стандартом для веб-приложений и обеспечивает необходимую безопасность.

Рассмотрим основные типы клиентов:

* «тонкий» клиент: клиент, который выполняет минимальные вычисления, а большая часть обработки происходит на сервере. Это позволяет легко обновлять и поддерживать приложение;
* «толстый» клиент: клиент, который выполняет большую часть обработки на своем устройстве. Это может привести к повышению производительности, но усложняет обновления и поддержку.

Для разработки приложения была выбрана двухзвенная архитектура клиент-сервер с использованием тонкого клиента, что обеспечит легкость в обновлении и поддержку приложения. Эта архитектура позволит эффективно масштабировать систему и обеспечить безопасность взаимодействия между пользователем и сервером.

* 1. Структурная схема системы

Система – это совокупность взаимосвязанных элементов, которые взаимодействуют для достижения общей цели или выполнения определенной функции.

Суть структурного подхода к проектированию системы заключается в ее разбиении на автоматизируемые функции. Система делится на функциональные подсистемы, которые затем подразделяются на подфункции, а те – на задачи и так далее. Этот процесс продолжается до уровня конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, где все компоненты взаимосвязаны. В случае разработки системы «снизу-вверх», начиная с отдельных задач и переходя к общей системе, целостность может быть утрачена, что приводит к проблемам с интеграцией отдельных компонентов.

Наиболее распространенные методологии структурного подхода основываются на нескольких общих принципах. Два ключевых принципа следующие:

* принцип «разделяй и властвуй» – это подход к решению сложных задач путем их деления на более мелкие, независимые задачи, которые проще понять и решить;
* принцип иерархического упорядочивания – это организация компонентов проблемы в иерархические структуры с добавлением новых деталей на каждом уровне.

Система должна представлять собой совокупность элементов, находящихся в определенной взаимосвязи и образующих единое целое, направленное на достижение конкретной цели. Она может быть частью более сложной системы и включать в себя более простые системы. Таким образом, систему можно рассматривать как набор подсистем, организованных для достижения определенной цели и описанных с помощью различных моделей, а подсистему – как группу элементов, часть которых отвечает за специфику поведения, представленного другими ее компонентами.

На рисунке 8 приведена структурная схема разрабатываемой системы.

В состав клиентской части входит:

1. подсистема взаимодействия с сервером, которая осуществляет установку соединения с сервером, формирование и отправку запросов;
2. справочная подсистема, которая отвечает за выдачу сведений о системе и ее разработчиках;
3. подсистема вывода данных, включает в себя подсистему визуализации, которая будет визуализировать работу с игровым полем;
4. подсистема ввода данных:

* подсистема «Ученик», которая включает в себя подсистему составления задания;
* подсистема регистрации и авторизации, которая позволит пользователям получать доступ к системе;
* подсистема «Учитель», которая состоит из:
* подсистемы работы с заданием, которая позволит создавать тексты заданий и игровые поля;
* подсистемы работы с журналом, которая позволит учителю оценивать работу учеников и просматривать их успеваемость;
* подсистемы работы с учеником, которая отвечает за управление данными ученика.

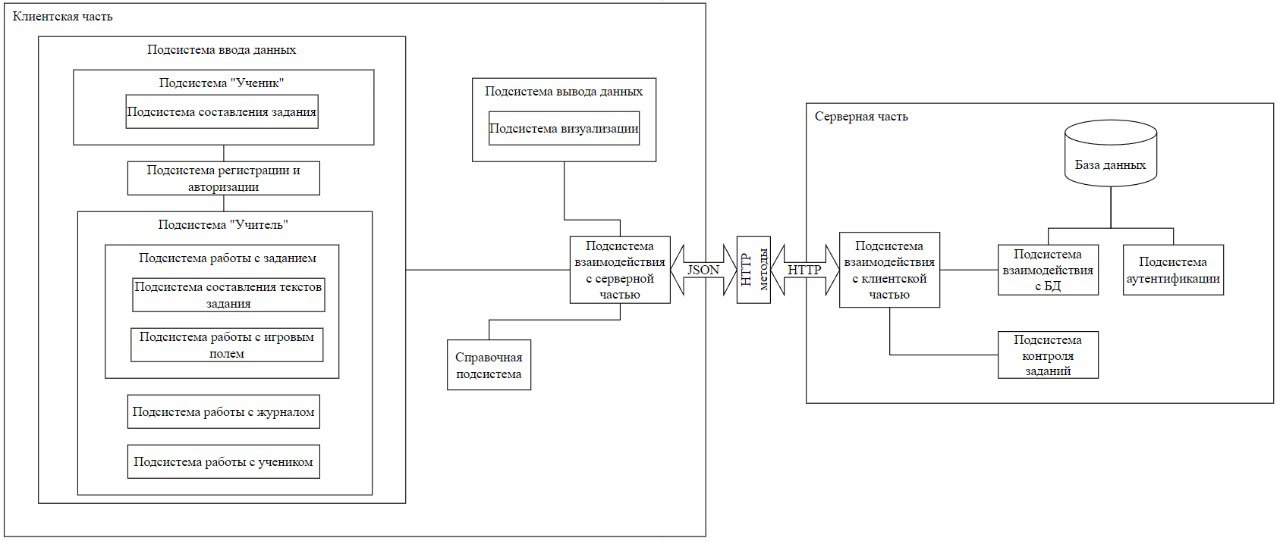
В состав серверной части входит:

1. база данных;
2. подсистема взаимодействия с базой данных, которая осуществляет доступ к базе данных и обеспечивает хранение и извлечение данных;
3. подсистема аутентификации, которая проверяет данные при входе в систему;
4. подсистема контроля заданий, которая проверяет правильность составленной учеником программы;
5. подсистема взаимодействия с клиентской частью, которая осуществляет приём данных от клиента и передает их на обработку серверу.
   1. Разработка спецификации требований

Спецификация требований к программному обеспечению представляет собой детальное описание системы, которую предстоит разработать [14]. Она строится по аналогии со спецификацией бизнес-требований, устанавливая функциональные и нефункциональные характеристики. В неё могут входить сценарии использования, описывающие взаимодействие пользователя с программой, чтобы обеспечить оптимальный пользовательский опыт.

Этот документ становится основой для соглашения между заказчиком и подрядчиком (или между отделами маркетинга и разработки в рыночно-ориентированных проектах) относительно ожидаемых функций программного продукта. Спецификация требований к программному обеспечению представляет собой строгую оценку потребностей, предваряющую этапы проектирования системы, и направлена на снижение необходимости последующих доработок. При правильной разработке такой документ помогает избежать срыва проекта.

Спецификация требований должна содержать перечень требований, достаточных для выполнения разработки. Для этого разработчику необходимо четко и полно представлять разрабатываемый продукт, что достигается через регулярное взаимодействие с командой проекта и заказчиком на всех этапах разработки программного обеспечения.

  
Рисунок 8 – Структурная схема системы (для АС «Обучающая система», технология «тонкий» клиент»)

* + 1. Функциональная спецификация

Функциональные требования определяют, какие функции будет выполнять разрабатываемая система [15]. Они описывают, каким образом система будет работать для удовлетворения потребностей пользователей и как она должна реагировать на определенные команды и действия, чтобы соответствовать ожиданиям.

Наиболее распространенные виды функциональных требований включают:

* бизнес-правила;
* требования к сертификации;
* требования к отчетности;
* административные функции;
* уровни авторизации;
* отслеживание аудита;
* внешние интерфейсы внешние интерфейсы;
* управление данными;
* законодательные и регуляторные требования.

Функциональная спецификация системы приведена в таблице 2.

* + 1. Перечень исключительных ситуаций

Исключительная ситуация – это ситуация, при которой система не может выполнить возложенных на нее функций или которая может привести к денормализации работы системы.

В таблице 4 приведен перечень исключительных ситуаций для разрабатываемой системы и описаны реакции системы на их возникновение.

Таблица 3 – Перечень функций, выполняемых системой

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название  подсистемы | Название функции | Информационная среда | | | |
| Входные данные | | Выходные данные | |
| Назначение (наименование) | Тип, ограничения | Назначение (наименование) | Тип, ограничения |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Справочная | Выдать сведения о разработчиках | Сведения о разработчиках системы (ФИО, номер группы) | Текст (МЕМО) | Визуальное отображение информации | – |
| Выдать сведения о системе | Сведение о системе | Текст (HTML) |
| Код ошибки | Целое |
| Регистрация и авторизация | Ввести логин | Набор допустимых символов | Латинские буквы, цифры, «\_», «.», «!», «@» | Логин | Строка |
| Допустимая длина | Целое  3..15 | Код ошибки | Целое |
| Ввести пароль | Набор допустимых символов | Латинские буквы, цифры, «\_», «.», «!», «@» | Пароль | Строка |
| Допустимая длина | Целое  8..15 | Код ошибки | Целое |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Регистрация и авторизация | Ввести повтор пароля | Набор допустимых символов | Латинские, русские буквы, цифры, «\_» | Повтор пароля | Строка |
| Допустимая длина | Целое 6..20 | Код ошибки | Целое |
| Работа с игровым полем | Ввести ширину игрового поля | Допустимый диапазон | Целое 4..10 | Ширина | Целое |
| Код ошибки | Целое |
| Ввести длину игрового поля | Допустимый диапазон | Целое 4..10 | Длина | Целое |
| Код ошибки | Целое |
| Ввести количество кубиков | Допустимый диапазон | Вещественное 0..15% от размеров игрового поля | Количество кубиков | Целое |
| Код ошибки | Целое |
| Ввести количество лунок | Допустимый диапазон | Вещественное 0..10% от размеров игрового поля | Количество лунок | Целое |
| Код ошибки | Целое |
| Работа с заданием | Выбрать уровень сложности | Список допустимых  значений | Массив строк | ID уровня сложности | Целое |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Составление текстов задания | Ввести название главы | Набор допустимых символов | Латинские буквы, цифры, «\_», «.», «!», «@», «,», « » | Название главы | Строка |
| Допустимая длина | Целое, 1..20 символов | Код ошибки | Целое |
| Ввести название уровня | Набор допустимых символов | Латинские буквы, цифры, «\_», «.», «!», «@» | Название уровня | Строка |
| Допустимая длина | Целое, 1..20 символов | Код ошибки | Целое |
| Ввести текст подсказки | Набор допустимых символов | Латинские буквы, цифры, «\_», «.», «!», «@» | Текст подсказки | Строка |
| Допустимая длина | Целое, 1..200 символов | Код ошибки | Целое |
| Установить срок сдачи | Набор допустимых символов | Цифры, «.» | Дата | Дата |
| Допустимая длина | 12 символов | Код ошибки | Целое |

Таблица 4 – Перечень исключительных ситуаций

|  |  |  |
| --- | --- | --- |
| Название  подсистемы | Название исключительной ситуации | Реакция системы |
| 1 Справочная | 1.1 Невозможно открыть файл справки | Выдача сообщения «Файл справки поврежден» |
| 1.2 Невозможно найти файл справки | Выдача сообщения «Отсутствует файл справки» |
| 2 Взаимодействие с БД | Попытка получения несуществующих данных | Выдача сообщения «Данные не найдены» |
| 3 Взаимодействие с серверной частью | Нестабильное соединение с интернетом | Выдача сообщения «Ошибка подключения» |

* 1. Разработка прототипа интерфейса пользователя системы

Интерфейс пользователя (User Interface) – это набор средств и элементов, с помощью которых пользователи взаимодействуют с системой, приложением или устройством. Он включает в себя визуальные компоненты (кнопки, меню, окна, иконки), текстовые элементы (подсказки, уведомления) и другие интерактивные элементы, которые помогают пользователю выполнять задачи и получать информацию [16].

Прототипирование – это процесс создания предварительной версии пользовательского интерфейса для приложения, веб – сайта или другого цифрового продукта. Прототип служит визуальным и интерактивным макетом, который демонстрирует структуру, функциональность и основные элементы будущего интерфейса [17].

На рисунке 9 приведен прототип страницы регистрации ученика в системе. Здесь пользователь должен ввести свои данные, которые в дальнейшем он будет использовать для повторного входа в систему. Пользователь будет вводить свою электронную почту, имя и придумывать пароль. При нажатии кнопки «Зарегистрироваться» система должна открыть главную страницу приложения. Также на странице должна располагаться гиперссылка для перехода на страницу авторизации пользователя.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание   
Рисунок 9 – Прототип страницы регистрации пользователя

На рисунке 10 представлен прототип страницы авторизации пользователя в системе, где он должен выбрать роль, под которой он будет работать в системе. На прототипе показан интерфейс для роли «Ученик». При авторизации «Ученик» должен ввести свою электронную почту, а также пароль. Если выбрана роль «Учитель», то требуется ввести только пароль. Также на странице должна располагаться гиперссылка для перехода на страницу регистрации пользователя.

Изображение выглядит как снимок экрана, текст, Графика, диаграмма

Автоматически созданное описание

Рисунок 10 – Прототип страницы авторизации пользователя

На рисунке 11 представлен прототип главной страницы веб-приложения. Здесь пользователь будет выполнять задания, сможет прочитать текст задания, написать к нему код, запустить его и увидеть, как он выполняется. В верхней части страницы должна располагаться кнопка, которая позволит открыть меню и выбрать задание, которое он будет выполнять. Помимо этого, здесь должна располагаться кнопка «Конструктор», для администратора, которая позволяет перейти к конструктору уровней. Также правее должна располагаться кнопка «Профиль», позволяющая пользователю перейти на страницу своего профиля. Ниже должны быть расположены кнопки «Подсказка» и «Учебник», при нажатии на кнопку «Подсказка» будет открыто окно, на котором будет описана подсказка к текущему уровню. Кнопка «Учебник» должна будет открывать окно, на котором будет описан список команд, доступных пользователю.

Изображение выглядит как снимок экрана, текст, Графическое программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 11 – Прототип главной страницы веб-приложения

На рисунке 12 представлен прототип главной страницы с открытым выпадающем окном выбора заданий. Здесь будут описаны названия глав и уровней. Пользователю будет доступно выбрать задание для выполнения. Также в нижней части будет доступна кнопка «Справка», которая позволит открыть окно с описанием сведений о разработчиках веб – приложения.

Изображение выглядит как текст, снимок экрана, Графическое программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 12 – Прототип главной страницы веб-приложения

На рисунке 13 представлен прототип страницы конструктора заданий для случая, когда администратор выбрал ручной способ создания игрового поля. Первоначально он должен выбрать способа создания игрового поля («Автоматически» и «Вручную»). Если будет выбран способ «Вручную», то администратору потребуется ввести параметры игрового поля, переместить шаблоны препятствий на игровое поля, выбрать уровень сложности и описать задание, написав название главы, уровня, текст задания и текст подсказки к заданию. В заключении формирования уровня, потребуется ввести «Дедлайн» - дату окончания выполнения задания и нажать кнопку «Сохранить». Также администратор сможет очистить все введенные параметры и игровое поле, нажав на кнопку «Очистить». В центре экрана расположена динамическая сетка игрового поля, которая будет изменяться в зависимости от введенных параметров.

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 13 – Прототип конструктора игрового поля с ручным способом создания игрового поля

На рисунке 14 представлен прототип страницы конструктора заданий для случая, когда администратор выбрал автоматический способ создания игрового поля. Администратору потребуется выбрать уровень сложности, описать название главы, название уровня, текст задания, текст подсказки и задать дедлайн задания. Аналогично, должны присутствовать кнопки «Сохранить» и «Очистить», позволяющие сохранить введенные данные и очистить их соответственно.

Изображение выглядит как снимок экрана, текст, диаграмма, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 14 – Прототип конструктора игрового поля с автоматическим способом создания игрового поля

На рисунке 15 представлен прототип страницы профиля пользователя системы. В левой части страницы должно быть расположено меню, с помощью которого можно будет перейти на страницы с деталями профиля и успеваемостью ученика. Также ученик сможет выйти из аккаунта нажав на кнопку «Выход». В центре экрана должно быть расположено поле с учетными данными ученика, с помощью которых, пользователь сможет сменить своё имя и пароль от аккаунта, для этого потребуется ввести новые данные и нажать кнопку «Сохранить».

На рисунке 16 представлен прототип страницы успеваемости пользователей системы. Здесь он сможет увидеть статус выполнения заданий, приведенных в таблице, заблокировать ученика, скачать его код, выставить ему оценку и отфильтровать данные в таблице, с помощью полей «Статус выполнения» и «Название уровня».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 15 – Прототип страницы деталей профиля пользователя

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 16 – Прототип страницы успеваемости пользователя

На рисунке 17 представлен прототип страницы журнала системы. Здесь администратор сможет увидеть статус выполнения заданий конкретным учеником, скачать написанный им код, а также отфильтровать данные с помощью полней «Статус выполнения», «Название уровня» и «ФИО ученика».

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 17 – Прототип страницы журнала системы

На рисунке 18 представлена навигационная модель для учителя, на рисунке 19 – ученика. На рисунках наглядно показаны переходы от одной страницы к другой, эти переходы будут осуществляться с помощью нажатия на специальные кнопки и гиперссылки.

* 1. Разработка информационно-логического проекта системы

В процессе разработки информационно-логического проекта системы проводится структурное проектирование, которое охватывает анализ требований, моделирование данных и процессов, а также определение функциональной и логической структуры системы [18].

Целью информационно-логического проектирования является создание детальной модели системы, которая отражает ее функциональные возможности, основные процессы, взаимодействия между компонентами и пользовательские требования. Использование информационно-логического подхода в проектировании позволяет создать наглядное представление системы, и выявить потенциальные проблемы на ранних стадиях разработки.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, дизайн

Автоматически созданное описание

Рисунок 18 – Навигационная модель для учителя

Изображение выглядит как снимок экрана, дизайн

Автоматически созданное описание

Рисунок 19 – Навигационная модель для ученика

1. Язык UML

UML (Unified Modeling Language) – это стандартизированный язык моделирования, предназначенный для создания схем и документации в процессе разработки программного обеспечения. Язык UML не является языком программирования. (он инвариантен к языкам программирования). Это графический язык, в котором каждой фигуре, символу, стрелке или их сочетаниям присвоены конкретные значения. Язык UML нужен, чтобы описать и визуализировать какую-то абстрактную модель. На практике это может быть:

* создание модели объекта. Например, описание структуры базы данных;
* создание модели процессов. Например, последовательность выполнения запросов ПО, чтобы клиент получил ожидаемый результат.

Схему на языке UML можно составить по существующему объекту или процессу либо создать на этапе проектирования, чтобы разрабатывать объект или отлаживать процесс. Диаграммы UML применяют в проектировании, презентациях, описании или создании документации [19].

* + 1. Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актерами и вариантами использования.

Для специфицирования (построения точных, недвусмысленных полных моделей) системы и ее документирования используется унифицированный язык моделирования UML. Она служит для описания функциональных требований к системе, позволяя увидеть, как различные пользователи (актеры) взаимодействуют с системой через определенные функции или сценарии, называемые вариантами использования [20].

На рисунке 20 приведена диаграмма вариантов использования (пользователя и ученика).

Изображение выглядит как рисунок, зарисовка, диаграмма, текст

Автоматически созданное описание

Рисунок 20 – Диаграмма вариантов использования системы актера «Пользователь» и «Ученик»

При авторизации в системе первоначально пользователь должен выбрать роль. Выбрав роль «Ученик», пользователь должен ввести email и пароль, а выбрав роль «Учитель», авторизация проходит только с помощью пароля. Ученик может зарегистрироваться в системе, для этого необходимо ввести имя, email, пароль и повтор пароля. Пользователь может посмотреть справочную информацию о разработчиках и системе, а также данные профиля и на своё усмотрение поменять своё имя и пароль. В системе предусмотрены две роли: ученик и учитель.

На рисунке 21 приведена диаграмма вариантов использования для ученика. Ученик может выполнить задание: выбрать его из списка в боковом меню, написать к нему код программы (путем ввода команд), запустить его, отправить на проверку или очистить написанный им код. При выполнении задания он может посмотреть подсказку к заданию и учебник. Также ученик может посмотреть свою успеваемость и настроить её отображение при помощи фильтров, задав их значения.

На рисунке 22 приведена диаграмма вариантов использования для учителя. Учитель может работать в конструкторе, где первоначально требуется выбрать способ создания игрового поля (ручной или автоматический). В ручном режиме он должен самостоятельно задать параметры игрового поля (длину и ширину, количество занятых клеток, кубиков и лунок). Также в данном режиме предусмотрена работа с игровым полем, посредством добавления и удаления шаблонов. Также, учитель должен выбрать уровень сложности из выпадающего списка (легкий, средний и сложный). Во время составления задания он также должен написать название главы и уровня, текст задания, подсказку к нему и задать дедлайн. В заключении работы с конструктором учитель должен сохранить сформированное задание, а также может очистить введенные параметры.

В ходе работы с журналом учитель может посмотреть таблицу успеваемости, в которой будет возможность заблокировать ученика, загрузить код его программы, выставить оценку (зачет, незачет) и применить фильтры для отображения интересующей информации.

Изображение выглядит как текст, рисунок, диаграмма, зарисовка

Автоматически созданное описание

Рисунок 21 – Диаграмма вариантов использования актера «Ученик»

Изображение выглядит как текст, диаграмма, Шрифт, карта

Автоматически созданное описание

Рисунок 22 – Диаграмма вариантов использования актера «Учитель»

В ходе проверки задания учитель может выбрать задание для выполнения и проверки корректности его составления. У него будет возможность написать код программы для запуска его на выполнение. Также учитель может проверить написанный учеником код программы.

* + 1. Сценарии

Сценарий (scenario) ‑ определенная последовательность действий, которая описывает действия актеров и поведение моделируемой системы в форме обычного текста [27].

В контексте языка UML сценарий используется для дополнительной иллюстрации взаимодействия актеров и вариантов использования.

Рассмотрим несколько сценариев.

Вариант использования: «Задать дедлайн»

Краткое описание. Позволяет учителю во время создания задания задать время окончания его выполнения. Включается в вариант использования «Работать в конструкторе».

Актант. Учитель.

Предусловия. Компьютер включен, вход в систему выполнен успешно. На экране показан интерфейс конструктора заданий.

Основной поток событий.

1. Система выводит на экран страницу конструктора заданий (см. рисунок 13). На странице отображаются кнопки «Конструктор», «Профиль», «Автоматически», «Вручную», «Сохранить», «Очистить». Также на странице отображаются неактивные поля ввода «Ширина», «Высота», «Количество кубиков», «Количество занятых клеток», «Название главы», «Название уровня», «Текст задания», «Текст подсказки», «Дедлайн», выпадающий список «Уровень сложности» и шаблоны игрового поля.
2. Учитель кликает левой кнопкой мыши (ЛКМ) по полю «Дедлайн» и вводит дату (в формате ДД.ММ.ГГГГ).

А1: Учитель щелкает по кнопке «Конструктор».

А2: Учитель щелкает по кнопке «Профиль».

А3: Учитель щелкает по кнопке «Автоматически».

А4: Учитель щелкает по кнопке «Вручную».

А5: Учитель щелкает по выпадающему списку «Уровень сложности».

А6: Учитель щелкает по одному из шаблонов игрового поля.

А7: Учитель щелкает по кнопке «Сохранить».

А8: Учитель щелкает по кнопке «Очистить».

А9: Учитель щелкает по клетке игрового поля.

1. Учитель нажимает на кнопку «Сохранить».
2. Система проверяет все поля на корректность по допустимому диапазону значений.

А10: Введенное значение некорректно.

1. Система сохраняет текущий уровень и появляется всплывающее окно, что задание успешно создано. Вариант использования заканчивается успешно.

Альтернативы.

А1: Учитель щелкает по кнопке «Конструктор».

А1.1 Система обновляет текущую страницу.

А1.2 Вариант использования завершается.

А2: Учитель щелкает по кнопке «Профиль».

А2.1 Переход к варианту использования «Посмотреть данные профиля».

А3: Учитель щелкает по кнопке «Автоматически».

А3.1 Переход к варианту использования «Создать автоматически».

А4: Учитель щелкает по кнопке «Вручную».

А4.1 Переход к варианту использования «Создать вручную».

А5: Учитель щелкает по выпадающему списку «Уровень сложности».

А5.1 Переход к варианту использования «Задать уровень сложности»

А6: Учитель щелкает по одному из шаблонов игрового поля.

А6.1 Переход к варианту использования «Выбрать шаблон».

А7: Учитель щелкает по кнопке «Сохранить».

А7.1 Переход к варианту использования «Сохранить задание».

А7.1.А1 Заполнены не все поля.

А7.1.А1.1 Система открывает модальное окно с требованием заполнить все текстовые поля. В окне также имеется кнопка «ОК».

А7.1.А1.2 Учитель читает сообщение и нажимает кнопку «ОК».

А7.1.А1.3 Система закрывает модальное окно.

А7.1.А2 Заполнены все поля.

А7.2 Вариант использования завершается.

А8: Учитель щелкает по кнопке «Очистить».

А8.1 Система очищает все заданные учителем параметры игрового поля.

А8.2 Вариант использования завершается.

А9: Учитель щелкает по клетке игрового поля.

А9.1 Переход к варианту использования «Выбрать место».

А10: Введенное значение некорректно.

А10.1 Система открывает модальное окно с сообщением об ошибке «Неверный формат данных». В окне также имеется кнопка «ОК».

А10.2 Учитель читает сообщение и нажимает кнопку «ОК»

А10.3 Система закрывает модальное окно.

А10.4 Переход к п. 2 основного потока.

Постусловия. При успешном завершении на экране появляется всплывающее окно с сообщением «Задание создано успешно».

Вариант использования: «Подтвердить смену пароля»

Краткое описание. Позволяет пользователю сменить пароль. Включается вариант использования «Посмотреть данные профиля».

Актант. Пользователь.

Предусловия. Компьютер включен, вход в систему выполнен успешно. Пользователь перешел в раздел «Профиль».

Основной поток событий.

1. Система выводит на экран страницу просмотра профиля (см. рисунок 15). На странице отображаются кнопки «Профиль», «Сохранить», «Успеваемость», «Выход». Также на странице отображаются неактивные поля ввода «Сменить имя», «Сменить пароль», «Подтвердить смену пароля».
2. Учитель щелкает левой кнопкой мыши (ЛКМ) по полю «Подтвердить смену пароля» и вводит новый пароль.

А1: Пользователь щелкает по кнопке «Выход».

А2: Пользователь щелкает по кнопке «Сохранить».

1. Пользователь нажимает на кнопку «Сохранить».
2. Система проверяет все поля на корректность.

А3: Поля не совпадают.

А4: Введенное значение некорректно.

1. Система сохраняет новый пароль пользователя в базу данных и появляется всплывающее окно, что пароль успешно сменен. Вариант использования заканчивается успешно.

Альтернативы.

А1: Пользователь щелкает по кнопке «Выход».

А1.1 Система совершает выход из аккаунта.

А1.2 Вариант использования завершается.

А2: Пользователь щелкает по кнопке «Сохранить».

А2.1 Переход к варианту использования «Сменить пароль».

А2.1.А1 Не было зафиксировано изменений.

А2.1.А1.1 Система открывает модальное окно с сообщением, о том, что не было зафиксировано изменений в полях. В окне также имеется кнопка «ОК».

А2.1.А1.2 Пользователь читает сообщение и нажимает кнопку «ОК».

А2.1.А1.3 Система закрывает модальное окно.

А2.1.А2 Изменения зафиксированы.

А2.2 Вариант использования завершается.

А3: Поля не совпадают.

А3.1 Переход к варианту использования «Сменить пароль».

А3.1.А1 Поля «Сменить пароль» и «Подтвердить смену пароля» не совпадают.

А3.1.А1.1 Система отображает модальное окно с сообщением, о том, что поля не совпадают. В окне также имеется кнопка «ОК».

А3.1.А1.2 Пользователь читает сообщение и нажимает кнопку «ОК».

А3.1.А1.3 Система закрывает модальное окно.

А5.1.А1.4 Переход к п.2 основного потока.

А3.1.А2 Поля «Сменить пароль» и «Подтвердить смену пароля» совпадают.

А3.2 Вариант использования завершается.

А4: Введенное значение некорректно.

А4.1 Система открывает модальное окно с сообщением об ошибке «Неверный формат данных». В окне также имеется кнопка «ОК».

А4.2 Пользователь читает сообщение и нажимает кнопку «ОК».

А4.3 Система закрывает модальное окно.

А4.4 Переход к п.2 основного потока.

Постусловия. При успешном завершении на экране всплывающее окно с сообщением «Пароль сменен успешно».

Вариант использования: «Применить фильтры»

Краткое описание. Позволяет ученику отобразить таблицу своей успеваемости применяя фильтры. Расширяет вариант использования «Посмотреть успеваемость».

Актант. Ученик.

Предусловия. Компьютер включен, вход в систему выполнен успешно. На экране показан интерфейс профиля.

Основной поток событий.

1. Система выводит на экран страницу просмотра успеваемости (см. рисунок 16). На странице отображаются кнопки «Профиль», «Фильтр», «Детали профиля», «Выход». Также на странице отображаются неактивное поле ввода «Название уровня» и выпадающий список «Статус выполнения».
2. Ученик щелкает ЛКМ по полю «Название уровня» и вводит интересующие его данные.

А1: Ученик щелкает по кнопке «Выход».

А2: Ученик щелкает по кнопке «Фильтр».

А3: Ученик щелкает ЛКМ по выпадающему списку «Статус выполнения».

1. Пользователь нажимает на кнопку «Фильтр».
2. Система проверяет наличие каких-либо данных в поле «Название уровня».

А4: Введенные значения фильтров не отвечают существующим в системе данным (список результатов пуст).

1. Система производит поиск по заданному параметру в БД данным, и выводит список результатов. Вариант использования заканчивается успешно.

Альтернативы.

А1: Пользователь щелкает по кнопке «Выход».

А1.1 Система совершает выход из аккаунта.

А1.2 Вариант использования завершается.

А2: Пользователь щелкает по кнопке «Фильтр».

А2.1 Поле «Название уровня» пусто.

А2.1.А1 Система не изменяет данные таблицы.

А2.1.А2 Переход к п.2 основного потока.

А2.2 Поле «Название уровня» заполнено.

А2.2.А1 Переход к п.4 основного потока.

А2.3 Вариант использования завершается.

А3: Ученик щелкает ЛКМ по выпадающему списку «Статус выполнения».

А3.1 Система открывает выпадающий список, в котором есть поля выбора: «любой», «зачет», «не зачет».

А3.2 Ученик выбирает оценку.

А3.3 Система запоминает выбранное значение.

А3.3 Переход к п.2 основного потока.

А4: Введенные значения фильтров не отвечают существующим в системе данным (список результатов пуст).

А4.1 Система открывает модальное окно с сообщением о том, что в системе не существует соответствующих данных. В окне также имеется кнопка «ОК».

А4.2 Учитель читает сообщение об ошибке и нажимает сообщение об ошибке и нажимает кнопку «ОК».

А4.3 Система закрывает модально окно.

А4.4 Переход к п.2 основного потока.

Постусловия. При успешном завершении на экране таблица успеваемости ученика с отфильтрованными данными.

Вариант использования: «Выбрать уровень сложности»

Краткое описание. Позволяет учителю во время создания задания задать уровень сложности игрового поля. Включается вариант использования «Работа в конструкторе».

Актант. Учитель.

Предусловия. Компьютер включен, вход в систему выполнен успешно. На экране показан интерфейс конструктора заданий.

Основной поток событий.

1. Система выводит на экран страницу конструктора заданий (см. рисунок 13). На странице отображаются кнопки «Конструктор», «Профиль», «Автоматически», «Вручную», «Сохранить», «Очистить». Также на странице отображаются неактивные поля ввода «Ширина», «Высота», «Количество кубиков», «Количество занятых клеток», «Название главы», «Название уровня», «Текст задания», «Текст подсказки», «Дедлайн», выпадающий список «Уровень сложности» и шаблоны игрового поля.
2. Учитель кликает ЛКМ по выпадающему списку «Выберите уровень сложности»
3. Система открывает выпадающий список с тремя вариантами выбора (Легкий, Средний, Сложный)
4. Учитель выбирает один из трех представленных вариантов

А1: Учитель щелкает по кнопке «Профиль».

А2: Учитель щелкает по кнопке «Автоматически».

А3: Учитель щелкает по кнопке «Вручную».

А4: Учитель щелкает по полю «Дедлайн».

А5: Учитель щелкает по кнопке «Сохранить».

1. Учитель нажимает на кнопку «Сохранить».
2. Система проверяет все поля на корректность по допустимому диапазону значений.

А6: Введенные значения некорректны.

1. Система сохраняет текущий уровень и появляется всплывающее окно, что задание успешно создано. Вариант использования заканчивается успешно.

Альтернативы.  
А1: Учитель щелкает по кнопке «Профиль».

А1.1 Переход к варианту использования «Посмотреть данные профиля».  
А2: Учитель щелкает по кнопке «Автоматически».

А2.1 Переход к варианту использования «Создать автоматически».

А3: Учитель щелкает по кнопке «Вручную».

А3.1 Переход к варианту использования «Создать вручную».

А4: Учитель щелкает по полю «Дедлайн».

А4.1 Переход к варианту использования «Задать дедлайн»

А5: Учитель щелкает по кнопке «Сохранить».

А5.1 Переход к варианту использования «Сохранить задание».

А5.1.А1 Заполнены все поля.

А5.2 Вариант использования завершается

А6: Введенные значения некорректны.

А6.1 Система открывает модальное окно с сообщением об ошибке. В окне также имеется кнопка «ОК».

А6.2 Учитель читает сообщение и нажимает кнопку «ОК».

А6.3 Система закрывает модальное окно.

А6.4 Переход к п.1 основного потока.

Постусловия. При успешном завершении на экране появляется всплывающее окно с сообщением «Задание создано успешно».

* + 1. Диаграмма классов

Диаграммы классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем, они показывают набор классов, интерфейсов и коопераций, а также их связи. На практике диаграммы классов применяют для моделирования статического представления системы, они служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения [21].

На рисунке 23 приведена диаграмма классов системы (этап проектирования).

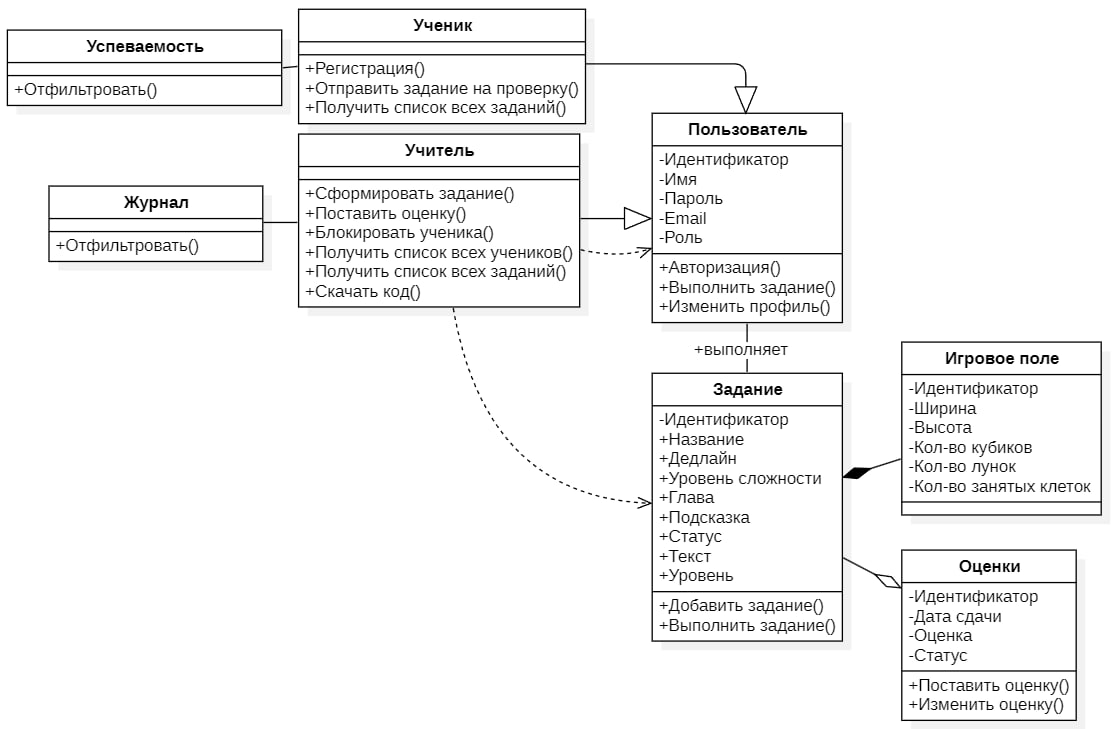


Рисунок 23 – Диаграмма классов системы

В таблице 5 приведено описание классов.

Таблица 5 – Описание классов системы

|  |  |
| --- | --- |
| Название класса | Назначение |
| 1 | 2 |
| Задание | Класс предназначен для хранения информации о задании. |
| Пользователь | Класс предназначен для хранения информации о пользователе системы. Он хранит в себе данные, такие как имя, фамилия, email, номер телефона и пароль. Также класс содержит информацию о роли пользователя, которая определяет его права и доступ в системе. |
| 1 | 2 |
| Игровое поле | Класс предназначен для хранения информации об игровом поле в системе. Игровое поле связано с определенным заданием. |
| Оценки | Класс предназначен для хранения информации об оценках. Используется для выставления оценок ученикам. |
| Учитель | Предоставляет функции для работы с БД. |
| Ученик | Реализует возможности необходимые для выполнения заданий. |

* + 1. Диаграмма состояний

В отличие от других диаграмм диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Именно для описания реакции объекта на подобные внешние воздействия и используются диаграммы состояний [28].

На рисунке 23 приведена диаграмма состояний системы.

Система подключается к базе данных. В случае если система не смогла установить соединение с БД, она переходит в состояние «Завершение работы с приложением». В случае успеха соединения система открывает страницу авторизации, где пользователю необходимо ввести логин и пароль, после чего система проверит учётные данные и в случае их корректности настроит интерфейс в соответствие с ролью пользователя. В случае некорректности введённых данных, пользователю предлагается ввести их повторно. Нажав на кнопку «Зарегистрироваться», система перейдёт в состояние регистрации, где пользователю необходимо будет ввести данные, после чего они будут проверены.

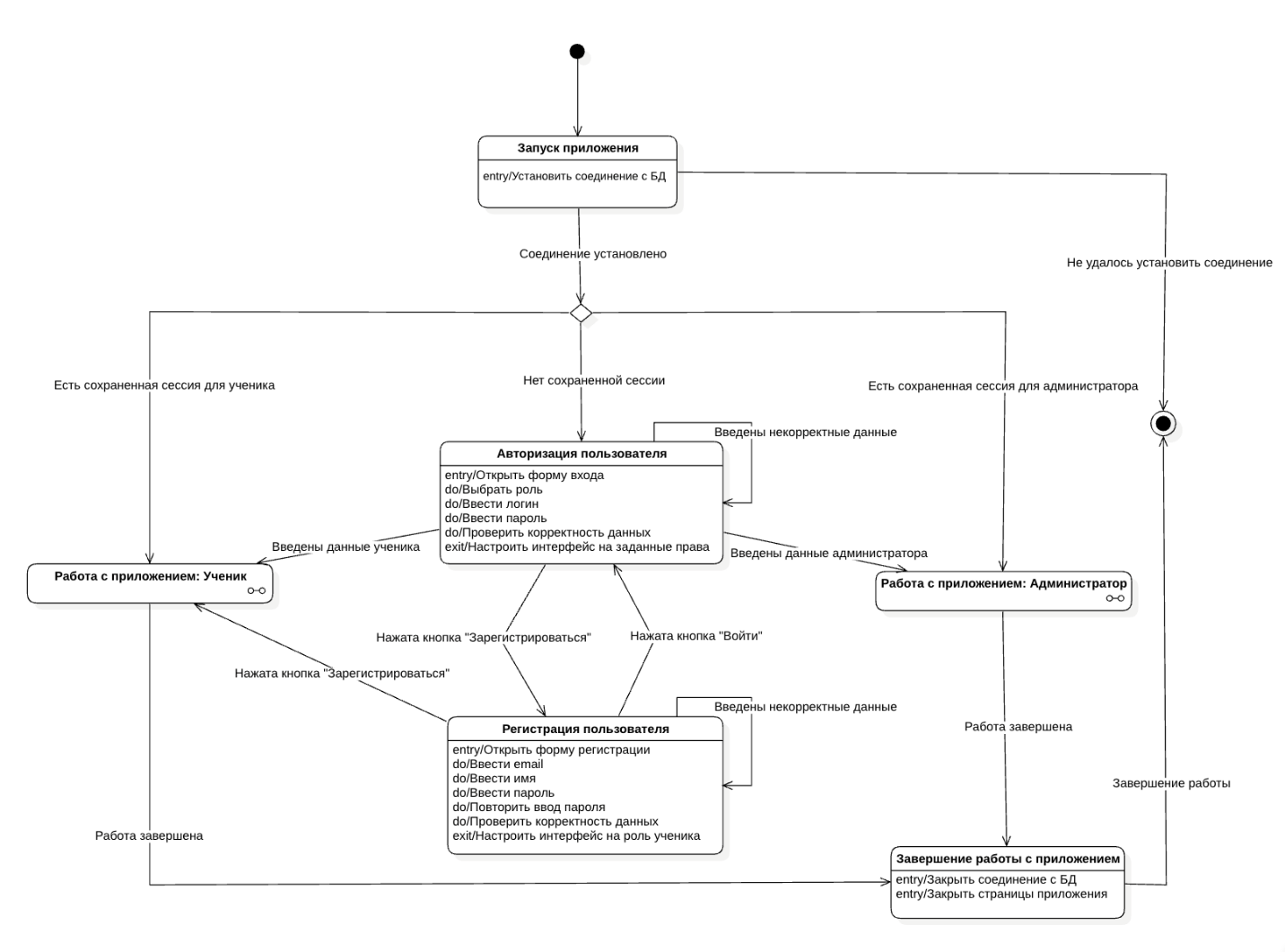


Рисунок 23 – Основная диаграмма состояний системы

Если данные корректны, то система зарегистрирует нового пользователя в системе, сохранив его данные. В случае некорректности введённых данных, пользователю предлагается ввести их повторно.

На рисунке 24 приведена диаграмма состояний системы для роли «Администратор».

По умолчанию система ожидает действий администратора. Система может перейти в состояние «О разработчиках», при нажатии на соответствующую кнопку. Нажатие на кнопку «Меню» откроет боковой список с глав, каждая из которых содержит выпадающий список заданий, после чего система, при нажатии на задание, перейдет в состояние «Просмотр задания». Также, администратор может открыть справку по нажатию соответствующей кнопки, и система перейдет в состояние «Просмотр справки». При выполнении задания, система перейдет в состояние «Отображение результатов выполнения задания», где администратору покажется всплывающее окно с результатами.

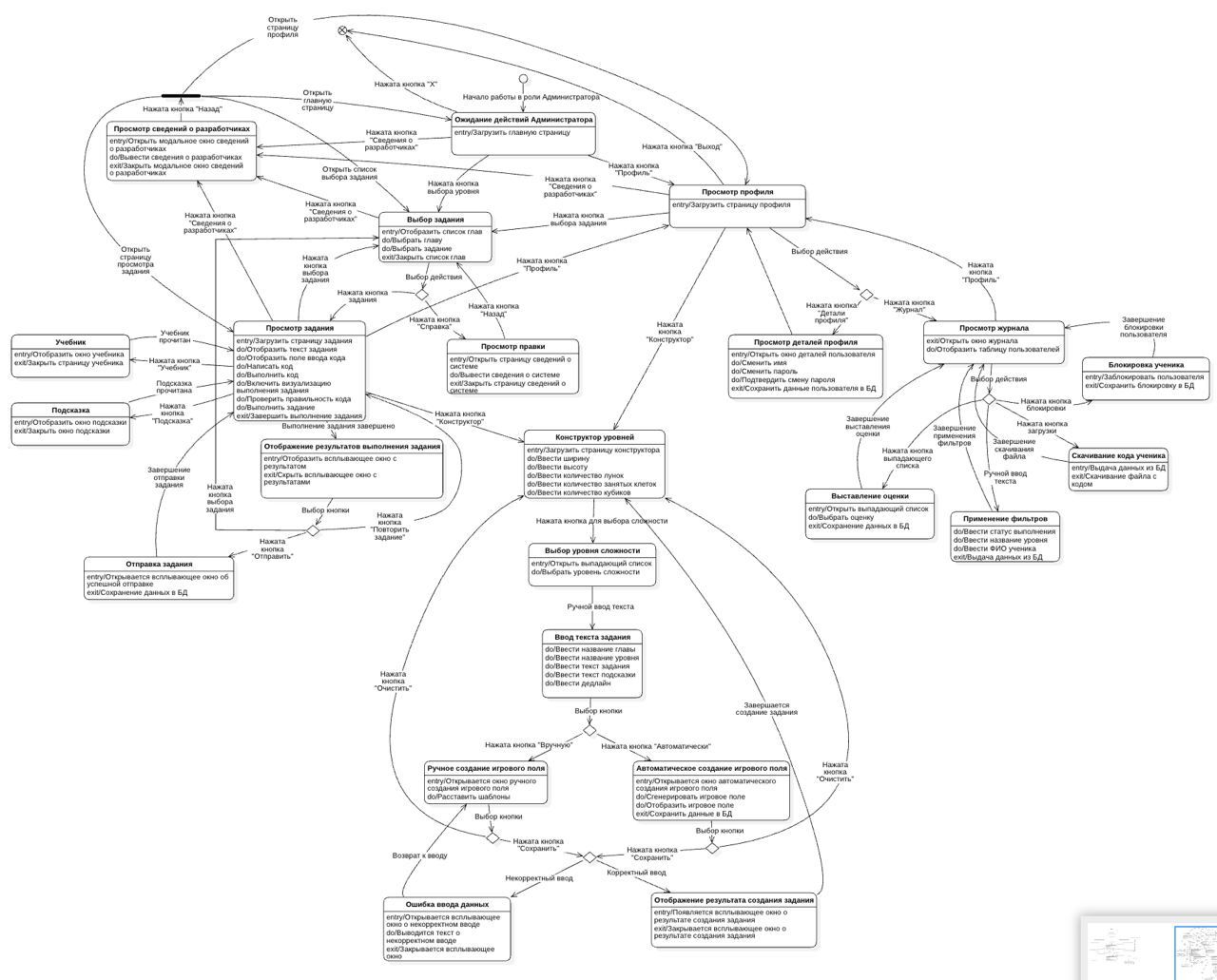


Рисунок 24 – Диаграмма состояний системы (администратор)

Если задание завершено успешно, то при нажатии на кнопку «Отправить» система перейдет в состояние «Отправка задания». Помимо этого, находясь на странице задания администратор может открыть окна «Учебник» и «Подсказка» и система перейдет в соответствующие состояния. Также, по кнопке «Профиль», администратор осуществляет переход в персональный профиль, и система переходит в состояние «Просмотр профиля». В профиле администратор может изменить личные данные включая пароль и имя. При нажатии на кнопку «Детали профиля» система перейдет в состояние «Просмотр деталей профиля». Помимо этого, администратор может открыть «Журнал» по соответствующей кнопке. Таким образом система переходит в состояние «Просмотр журнала». Находясь на странице журнала, администратор может применить фильтры для отображения информации об учениках. У администратора есть возможность скачать код ученика, заблокировать и выставить оценку. Эти действия соответствуют состояниям «Скачивание кода ученика», «Блокировка ученика» и «Выставление оценки». При нажатии на кнопку «Конструктор», открывается страница составления заданий, и система переходит в состояние «Конструктор уровней». На странице предлагается выбрать уровень сложности из выпадающего списка, и система переходит в состояние «Выбор уровня сложности». Затем требуется ввести название задания, главу, текст задания, подсказку и дедлайн. При вводе этих параметров система переходит в состояние «Ввод текста задания». Если администратор нажал на кнопку «Вручную», то система перейдет в состояние «Ручное создание игрового поля». Если администратор нажал на кнопку «Автоматически», то система перейдет в состояние «Автоматическое создание игрового поля». В случае, если данные были введены некорректно система переходит в состояние «Ошибка ввода данных», в противном случае система перейдет в состояние «Отображение результата создания задания».

На рисунке 25 приведена диаграмма состояний системы для роли «Ученик». По умолчанию система ожидает действий ученика. Система может перейти в состояние «О разработчиках», при нажатии на соответствующую кнопку. Нажатие на кнопку «Меню» откроет боковой список с глав, каждая из которых содержит выпадающий список заданий, после чего система, при нажатии на задание, перейдет в состояние «Просмотр задания». Также, ученик может открыть справку по нажатию соответствующей кнопки, и система перейдет в состояние «Просмотр справки». При выполнении задания, система перейдет в состояние «Отображение результатов выполнения задания», где ученику покажется всплывающее окно с результатами. Если задание завершено успешно, то при нажатии на кнопку «Отправить» система перейдет в состояние «Отправка задания». Помимо этого, находясь на странице задания администратор может открыть окна «Учебник» и «Подсказка» и система перейдет в соответствующие состояния. Также, по кнопке «Профиль», ученик осуществляет переход в персональный профиль, и система переходит в состояние «Просмотр профиля».

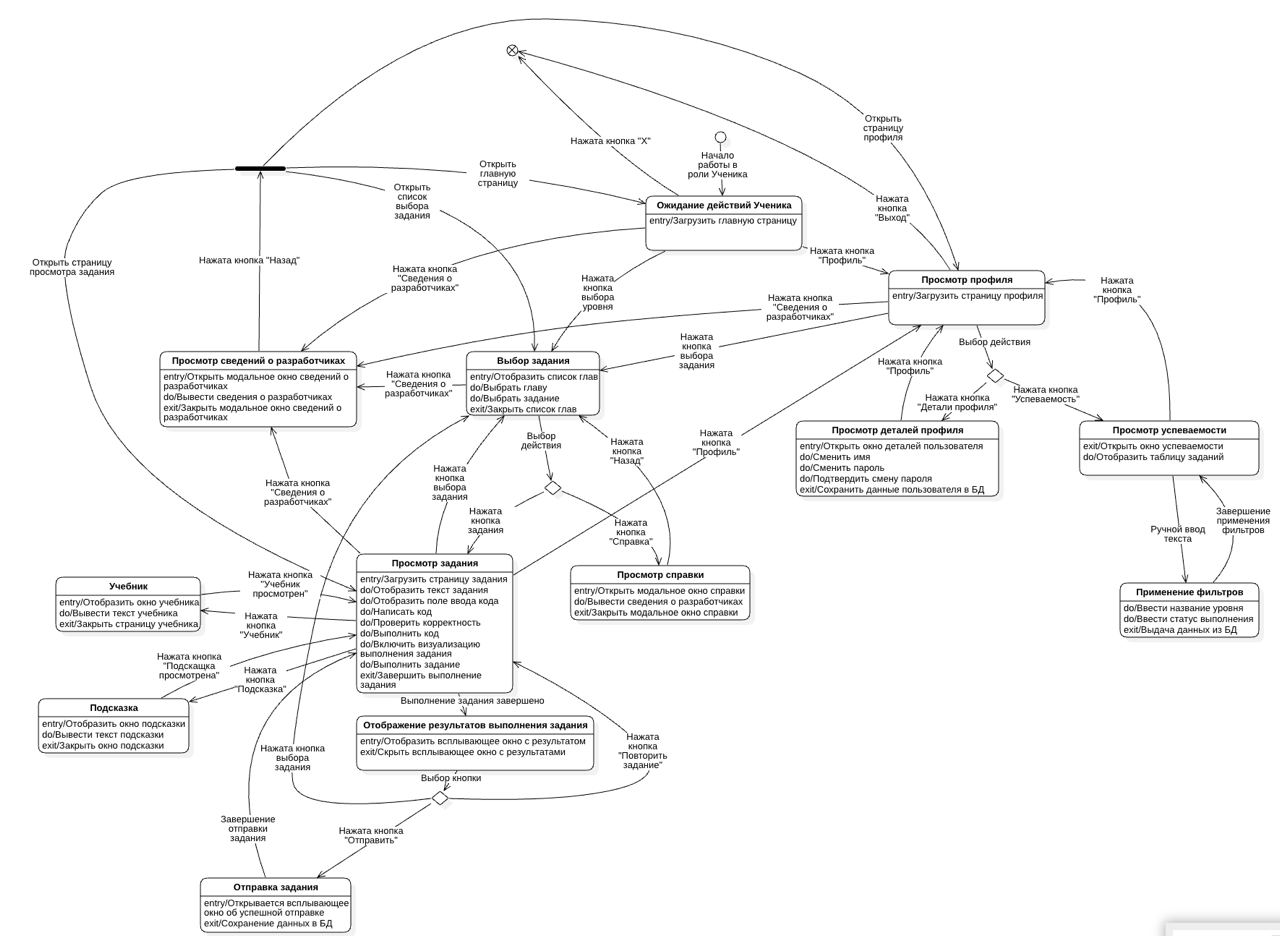


Рисунок 25 – Диаграмма состояний системы (ученик)

В профиле ученик может изменить личные данные, включая пароль и имя. При нажатии на кнопку «Детали профиля» система перейдет в состояние «Просмотр деталей профиля». Помимо этого, ученик может просмотреть «Успеваемость», нажав на соответствующую кнопку. Таким образом система переходит в состояние «Просмотр успеваемости». Находясь на странице успеваемости, ученик может применить фильтры для отображения информации о списке его заданий.

* + 1. Диаграмма деятельности

Диаграмма деятельности − это блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Результат может привести к изменению состояния системы или возвращению некоторого значения [29].

На рисунке 26 приведена диаграмма деятельности для работы с ручным созданием игрового поля. Система открывает страницу ручного создания игрового поля, где учитель может ввести параметры игрового поля, выбрать шаблон и нажать на клетку игрового поля, чтобы отобразить шаблон на нем. Также учитель может выбрать уровень сложности и ввести текст задания. Затем он нажимает кнопку «Сохранить» и система производит проверку введенных данных.

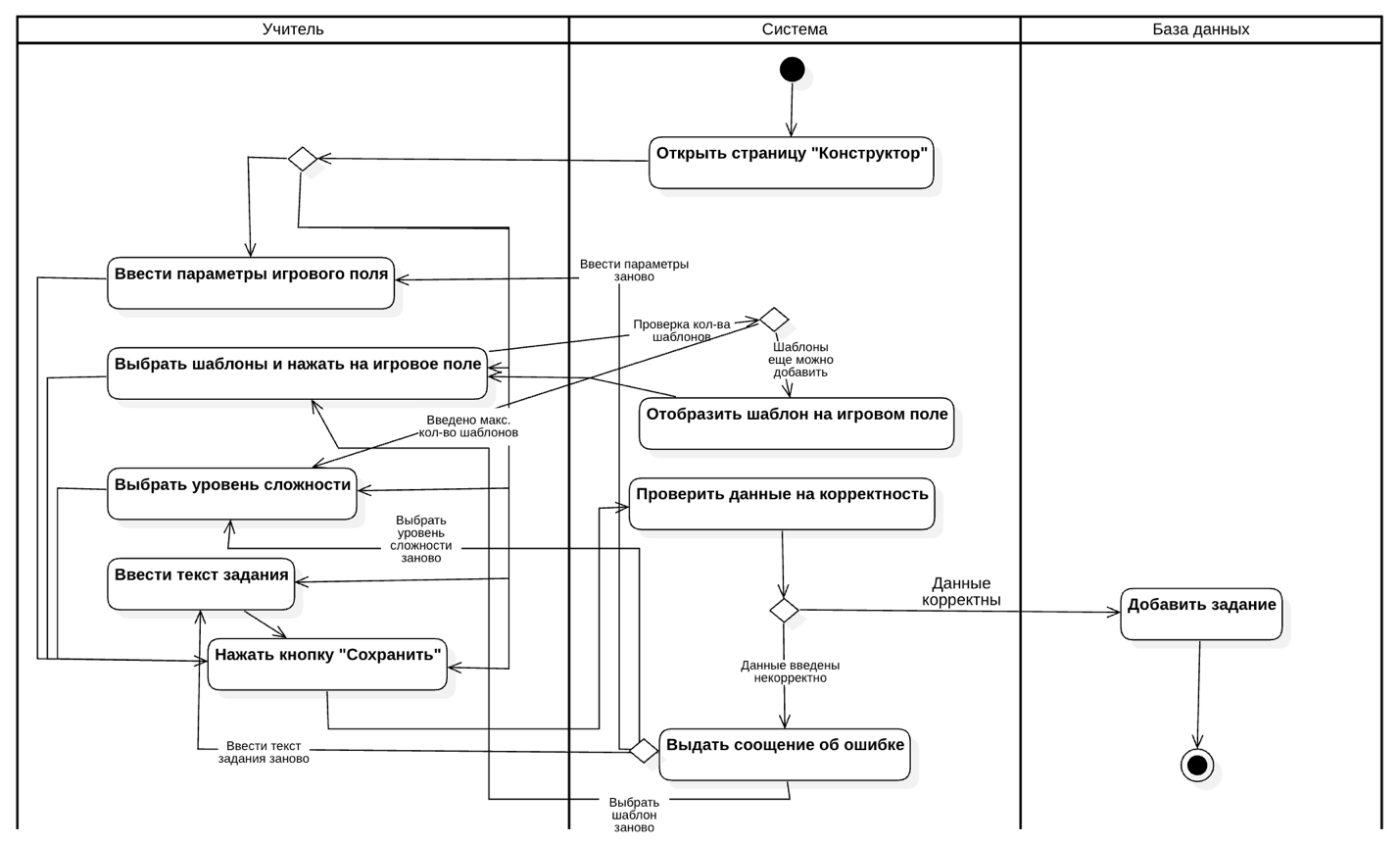


Рисунок 26 – Диаграмма деятельности работы с ручным созданием задания

Если данные корректны, то задание добавляется в БД, в противном случае система выводит сообщение об ошибке и возвращает учителя на повторный ввод данных.

* + 1. Диаграмма последовательности

Диаграмма последовательности − диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления [22]. Основными элементами диаграммы последовательностей являются обозначения объектов, вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

На рисунках 27 − 30 приведены диаграммы последовательности системы для вариантов использования «Задать дедлайн», «Подтвердить смену пароля», «Применить фильтры» и «Выбрать уровень сложности» соответственно. Диаграммы построены на основании сценариев, приведённых в п.2.5.4.



Рисунок 27 – Диаграмма последовательности «Выбрать уровень сложности»

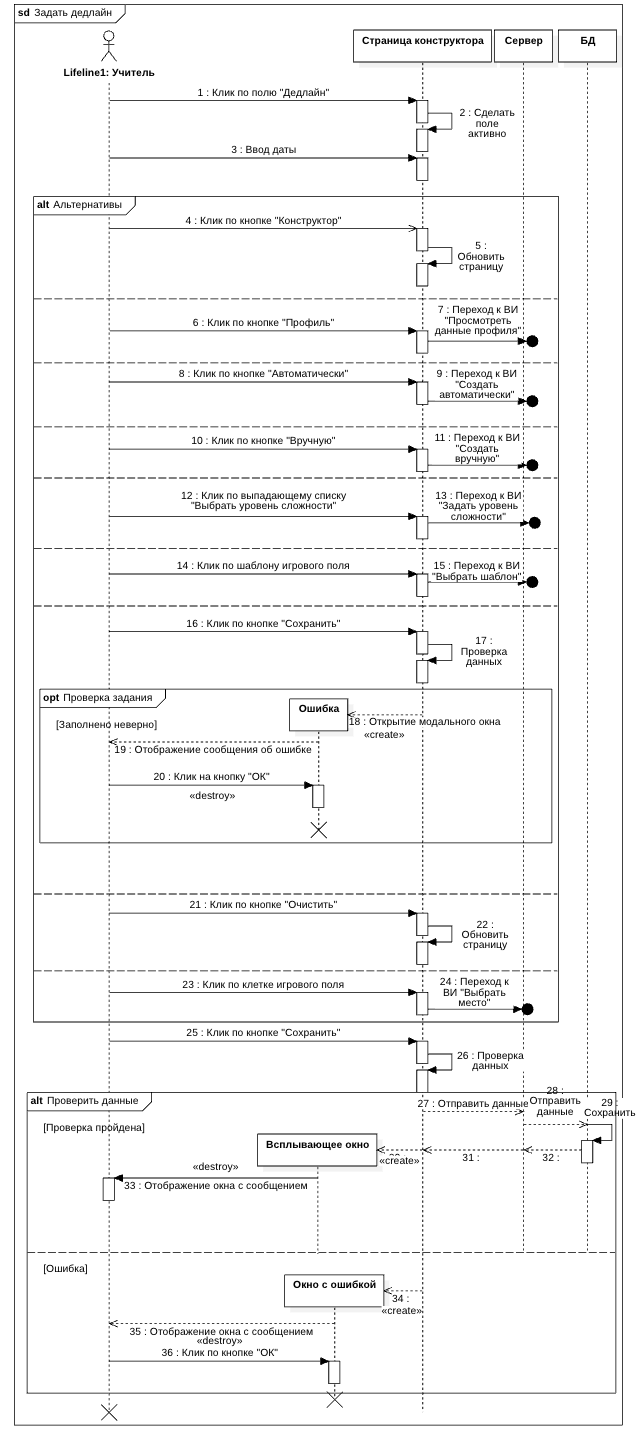


Рисунок 28 – Диаграмма последовательности «Задать дедлайн»

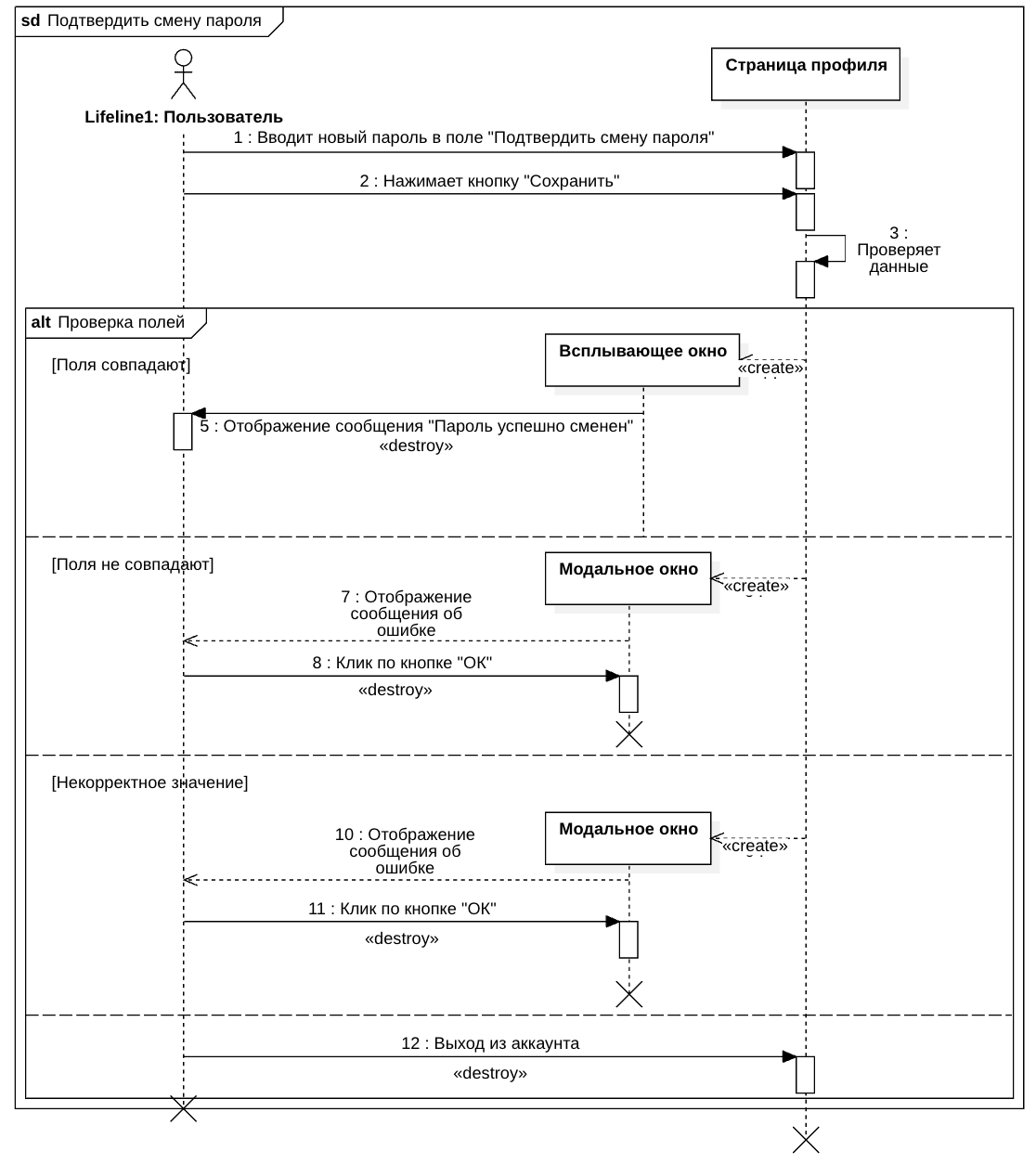


Рисунок 29 – Диаграмма последовательности «Подтвердить смену пароля»

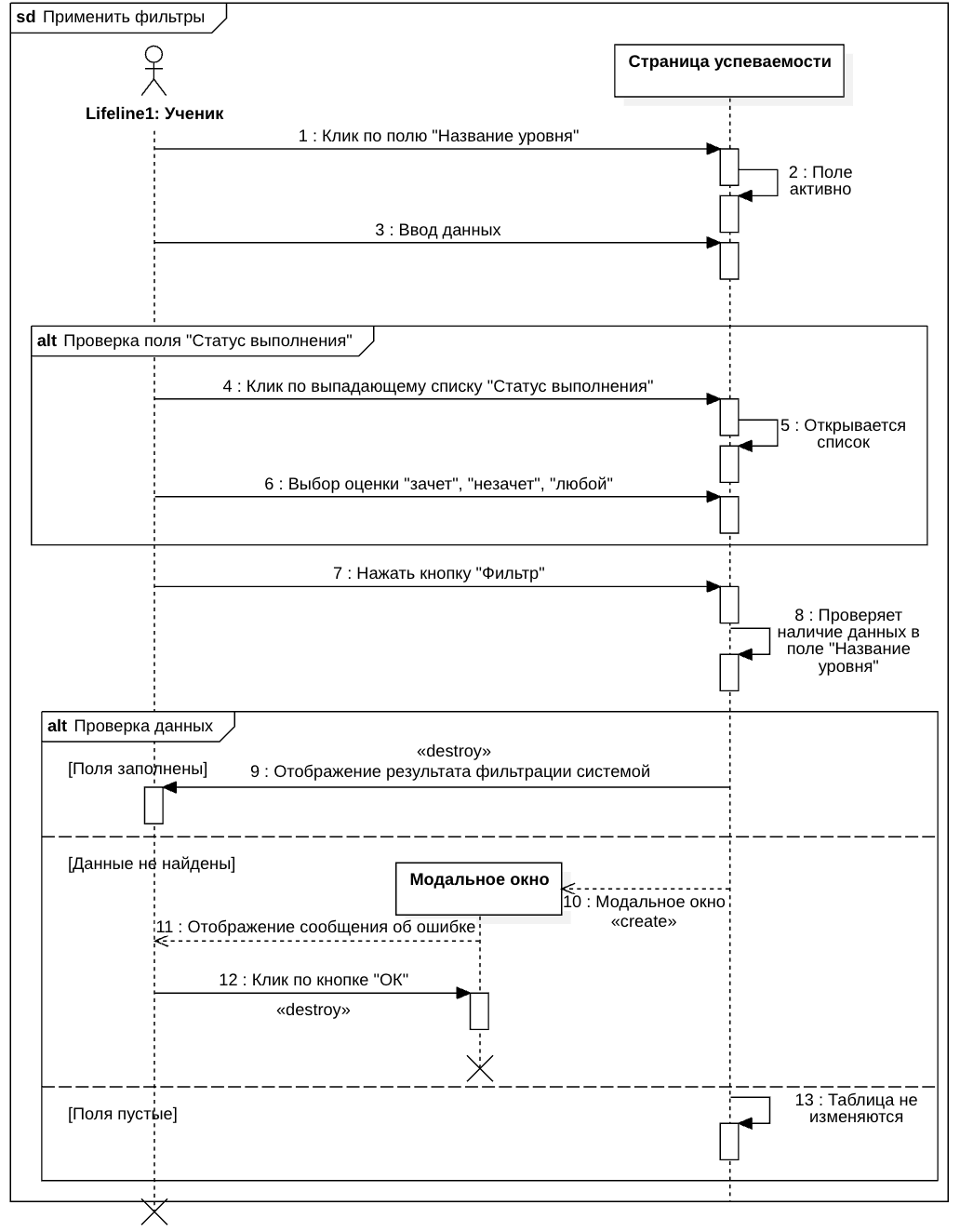


Рисунок 30 – Диаграмма последовательности «Применить фильтры»

* 1. Логическая модель данных

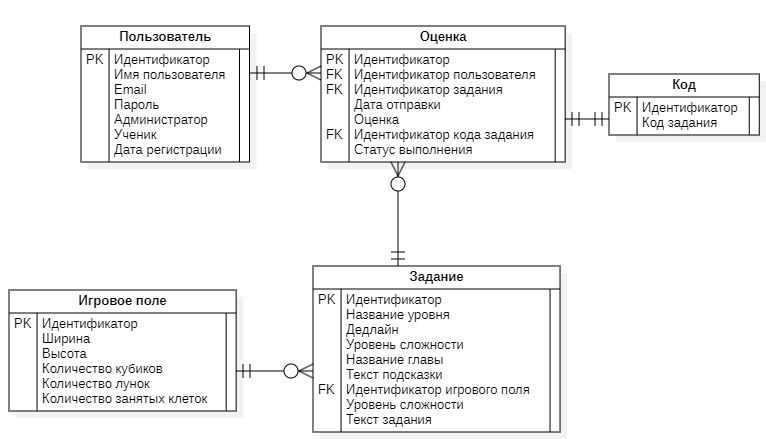
Логическая информационная модель – модель данных, в которой учитывается способ логического хранения данных в памяти ЭВМ. При построении модели базы данных (БД) используются следующие понятия.

Сущность – объект предметной области, который можно отличить от других понятий по некоторым признакам. Сущность состоит из множества своих экземпляров. Каждая сущность обладает свойствами – атрибутами [22]

Атрибут – определенное свойство сущности. Именно набор атрибутов, в общем случае уникальный для каждой сущности, позволяет выделить ее среди других объектов и назвать уникальным именем.

Атрибут или набор атрибутов, используемый для идентификации экземпляра сущности, называется ключом сущности. В случае если для идентификации экземпляра используется один атрибут, ключ называется простым; в противном случае ключ составной. Каждый экземпляр сущности однозначно определяется ключом [23]

Логическая модель БД разрабатываемой системы приведена на рисунке 31.

  
Рисунок 31 – Логическая модель данных

Описание объектов рассматриваемой предметной области, которые хранятся в базе данных, приведено в таблицах 5-8.

Таблица 6 – Сущность «Пользователь»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид пользователя | Целый | Уникальный идентификатор пользователя |
| Имя | Символьный[30] | Имя, используемое при идентификации пользователя и его взаимодействии с системой |
| Пароль | Символьный[15] | Пароль пользователя, преобразованный в закодированную строку |
| Логин | Символьный[15] | Электронная почта, указанная пользователем при регистрации |
| Роль | Символьный[7] | Роль пользователя, определяющая его роль в системе |
| Персонал | Логический | Указывает, является ли пользователь учителем или нет |
| Активный | Логический | Указывает, является ли пользователь активным или нет |
| Дата добавления | Дата | Дата добавления пользователя в систему |

Таблица 6 – Сущность «Оценки»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| 1 | 2 | 3 |
| Ид оценки | Целый | Уникальный идентификатор оценки |
| Ид пользователя | Целый | Уникальный идентификатор пользователя |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Ид задания | Целый | Уникальный идентификатор задания |
| Дата сдачи | Дата и время | Дата и время сдачи задания на проверку |
| Оценка | Перечисление | Оценка за задание |
| Ид кода | Целый | Указывает, является ли пользователь учителем или нет |
| Статус | Перечисление | Статус выполнения задания |

Таблица 7 – Сущность «Задание»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид задания | Целый | Уникальный идентификатор оценки |
| Название | Символьный | Название уровня |
| Дедлайн | Дата и время | Дата и время, до которого нужно сдать задание |
| Уровень | Целый | Номер уровня |
| Глава | Целый | Номер главы |
| Подсказка | Символьный[100] | Текст подсказки к заданию |
| Ид игрового поля | Целый | Уникальный идентификатор игрового поля |
| Сложность | Перечисление | Сложность, указанная в задании |
| Текст задания | Символьный[100] | Текст для выполнения задания |

Таблица 8 – Сущность «Игровое поле»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| 1 | 2 | 3 |
| Ид игрового поля | Целый | Уникальный идентификатор игорового поля |
| Ширина | Целый | Ширина игрового поля |
| Высота | Целый | Высота игрового поля |
| 1 | 2 | 3 |
| Количество кубиков | Целый | Количество кубиков на игровом поле |
| Количество лунок | Целый | Количество лунок на игровом поле |
| Количество занятых клеток | Целый | Количество занятых клеток на игровом поле |
| Поле JSON | JSON | JSON-формат расстановки и типов шаблонов игрового поля |

Таблица 9 – Сущность «Код»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| Ид код | Целый | Уникальный идентификатор кода |
| Код | Текстовый | Код программы |

* 1. Выбор и обоснование комплекса программных средств

Программные средства, используемые при разработке веб-приложений, представляют собой различные инструменты и технологии, которые помогают разработчикам создавать, тестировать, оптимизировать и поддерживать веб-приложения. Они включают в себя средства разработки, фреймворки и библиотеки, системы контроля версий, системы для работы с базами данных и многое другое.

Большинство средств предоставляет готовые решения для часто встречающихся задач (например, создание интерфейсов, управление базами данных), что позволяет разработчикам сосредоточиться на реализации уникальных аспектов приложения.

Средства для управления проектами, контроля версий и тестирования делают совместную работу более организованной, предотвращая конфликтные изменения и повышая продуктивность.

Программные средства – это не просто набор инструментов, а неотъемлемая часть всего процесса разработки, которая помогает снизить ошибки, ускорить реализацию и обеспечить высокое качество конечного продукта.

* + 1. Выбор языка программирования

В процессе разработки приложения был выбран язык программирования Python и фреймворк Django для серверной части и JavaScript для клиентской части.

Python – это высокоуровневый язык программирования общего назначения, который был разработан Гвидо ван Россумом и впервые представлен в 1991 году. Он отличается простым и понятным синтаксисом, что делает его доступным как для начинающих, так и для опытных разработчиков. Python широко используется в разработке веб-приложений, анализе данных, машинном обучении, искусственном интеллекте, автоматизации и многом другом.

Особенности Python:

1. читаемость и простота синтаксиса: Python ориентирован на улучшение читаемости кода, что упрощает разработку и сопровождение приложений.
2. кроссплатформенность: Python работает на всех основных операционных системах, включая Windows, macOS и Linux.
3. динамическая типизация: Python не требует указания типов данных, что ускоряет процесс разработки.
4. интерпретируемость: Код Python выполняется интерпретатором, что облегчает тестирование и отладку.
5. поддержка объектно-ориентированного программирования: Python поддерживает ООП, позволяя создавать масштабируемые и модульные приложения.

JavaScript – это высокоуровневый, интерпретируемый язык программирования, который был разработан Бренданом Эйхом и впервые представлен в 1995 году. Он стал основным языком для создания интерактивных и динамичных веб-страниц. JavaScript широко используется как на стороне клиента, так и на стороне сервера благодаря своей гибкости и большому числу фреймворков и библиотек.

Особенности JavaScript:

1. интерактивность и работа в браузере: JavaScript — единственный язык программирования, который изначально поддерживается всеми веб-браузерами без дополнительных настроек. Это делает его ключевым инструментом для создания интерактивных пользовательских интерфейсов, включая анимации, обработку событий и работу с DOM (Document Object Model).
2. асинхронность и работа с сетевыми запросами: JavaScript поддерживает асинхронное программирование через механизмы Promises, async/await и callbacks. Это позволяет эффективно работать с API, отправлять и обрабатывать сетевые запросы без блокировки основного потока выполнения.
3. кроссплатформенность: JavaScript может выполняться на разных устройствах и платформах. Например, благодаря Node.js он используется для серверной разработки, а такие технологии, как React Native, позволяют создавать мобильные приложения.
4. динамическая типизация: JavaScript является языком с динамической типизацией, что упрощает разработку и делает его более гибким. Тип переменной может меняться в процессе выполнения программы.
   * 1. Выбор фреймворков

Django – это мощный и масштабируемый веб-фреймворк на языке Python, созданный для упрощения и ускорения разработки веб-приложений. Django следует принципам "Don't Repeat Yourself" (DRY) и "The Web framework for perfectionists with deadlines", что подчеркивает его стремление к эффективному использованию времени и ресурсов разработчика.

Особенности Django:

* 1. модульность и масштабируемость: Django поддерживает модульную архитектуру, что позволяет разработчикам легко добавлять или изменять функциональность приложения.
  2. ОРМ (Объектно-реляционное отображение): Django включает встроенный инструмент ORM, который позволяет работать с базами данных (например, PostgreSQL, MySQL, SQLite) без написания SQL-запросов, что упрощает взаимодействие с данными.
  3. автоматическая генерация административной панели: Django автоматически создает административный интерфейс для управления данными, что значительно ускоряет разработку.
  4. поддержка безопасности: Django предоставляет встроенные механизмы защиты от распространенных угроз, таких как CSRF, SQL-инъекции и XSS-атаки.
  5. масштабируемость: Django подходит как для небольших проектов, так и для высоконагруженных систем благодаря встроенным возможностям работы с кэшем, подключению внешних сервисов и поддержке многопоточности.

Python и Django вместе предоставляют мощный набор инструментов для быстрой и эффективной разработки веб-приложений, минимизируя затраты на настройку и инфраструктуру.

* + 1. Выбор среды программирования

В качестве среды разработки была выбрана Visual Studio Code.

Visual Studio Code (VS Code) – это легкий, но мощный редактор кода, разработанный компанией Microsoft. Он поддерживает множество языков программирования и предоставляет расширенные возможности для разработки, отладки и управления проектами. VS Code является популярным инструментом среди веб-разработчиков благодаря своей гибкости и большому количеству доступных расширений.

Особенности Visual Studio Code:

* 1. расширяемость и поддержка расширений: VS Code имеет обширный рынок расширений, который позволяет пользователям добавлять новые функции и инструменты для работы с различными языками и фреймворками, такими как JavaScript, TypeScript, HTML, CSS и многие другие.
  2. интеграция с системами контроля версий: VS Code предоставляет встроенную поддержку Git и других систем контроля версий, что упрощает управление кодом и совместную работу над проектами. Разработчики могут легко отслеживать изменения, создавать ветки и выполнять коммиты прямо из редактора.
  3. отладка и интегрированные инструменты: VS Code включает мощные инструменты отладки, которые позволяют разработчикам эффективно тестировать и устранять ошибки в своем коде. Интеграция с браузерами и возможность работы с Live Server обеспечивают быстрый цикл разработки и тестирования веб-приложений.
  4. функция IntelliSense в VS Code предоставляет интеллектуальные подсказки по коду, включая автозаполнение, информацию о параметрах функций и документацию. Это значительно ускоряет процесс написания кода и помогает избежать ошибок, что особенно полезно при работе с большими проектами.
     1. Выбор операционной системы

Windows – это семейство операционных систем, разработанных компанией Microsoft [24]. В контексте выбора операционной системы, Windows является одной из самых популярных платформ благодаря своей распространенности, удобству в использовании и совместимости с большим количеством программного обеспечения и аппаратных решений.

Преимущества Windows 10:

1. совместимость с программным обеспечением: Windows 10 поддерживает огромный выбор программ и игр, что делает ее универсальной для большинства пользователей;
2. поддержка современных технологий: ОС поддерживает новейшие технологии, такие как виртуальная реальность, улучшенные графические интерфейсы и т. д.;
3. стабильность и производительность: Постоянные обновления и исправления багов делают Windows 10 одной из самых стабильных версий ОС.
   * 1. Выбор системы управления базами данных

PostgreSQL – это мощная объектно-реляционная СУБД с открытым исходным кодом, поддерживающая стандарт SQL и расширенные функции, такие как работа с JSON и геоданными. Она известна своей надежностью, масштабируемостью и гибкостью.

PostgreSQL применяется:

1. веб-приложения: благодаря поддержке сложных данных и интеграции с фреймворками (Django, Laravel);
2. анализ данных: удобна для больших объемов данных и работы с геоинформацией (PostGIS);
3. финансы: соответствует требованиям ACID, что делает её идеальной для банковских и бухгалтерских систем.

PostgreSQL востребована за универсальность и надежность в самых разных сферах.

1. Реализация системы
   1. Разработка и описание интерфейса пользователя

Рассмотрим основные особенности разработки интерфейса [25]:

* пользовательский опыт: основное внимание должно уделяться потребностям и предпочтениям пользователей. Исследования и тестирование с участием пользователей помогают создать более удобный и интуитивный интерфейс;

− удобство использования: интерфейс должен быть простым и понятным. Элементы управления должны быть логичными и легко доступными, чтобы пользователи могли быстро выполнять необходимые действия;

− консистентность: все элементы интерфейса должны быть согласованы по стилю, цветам и поведению. Консистентность помогает пользователям предсказать, как будут работать элементы, и облегчает обучение;

* адаптивность: интерфейс должен хорошо выглядеть и функционировать на различных устройствах и экранах (мобильные телефоны, планшеты, настольные компьютеры);

− эстетика: визуальный дизайн должен быть привлекательным и соответствовать бренду. Хорошо подобранные шрифты, цвета и графические элементы улучшают восприятие интерфейса;

− доступность: интерфейс должен быть доступен для людей с ограниченными возможностями. Это включает в себя использование контрастных цветов, доступность с помощью клавиатуры и поддержку экранных считывателей;

− тестирование и итерации: регулярное тестирование интерфейса и внесение изменений на основе отзывов пользователей – ключ к улучшению дизайна.

При запуске приложения открывается главная страница, которая приведена на рисунке 32. На этой странице пользователю открыт интерфейс вводного задания с игровым полем. На странице есть кнопки «Учебник» и «Подсказка», которые открывают модальные окна с соответствующей информацией. Также на странице есть кнопка «Профиль», по нажатию на которую откроется страница личного профиля пользователя. При нажатии на кнопку «Меню», пользователь может выбрать другие задания на выполнение.

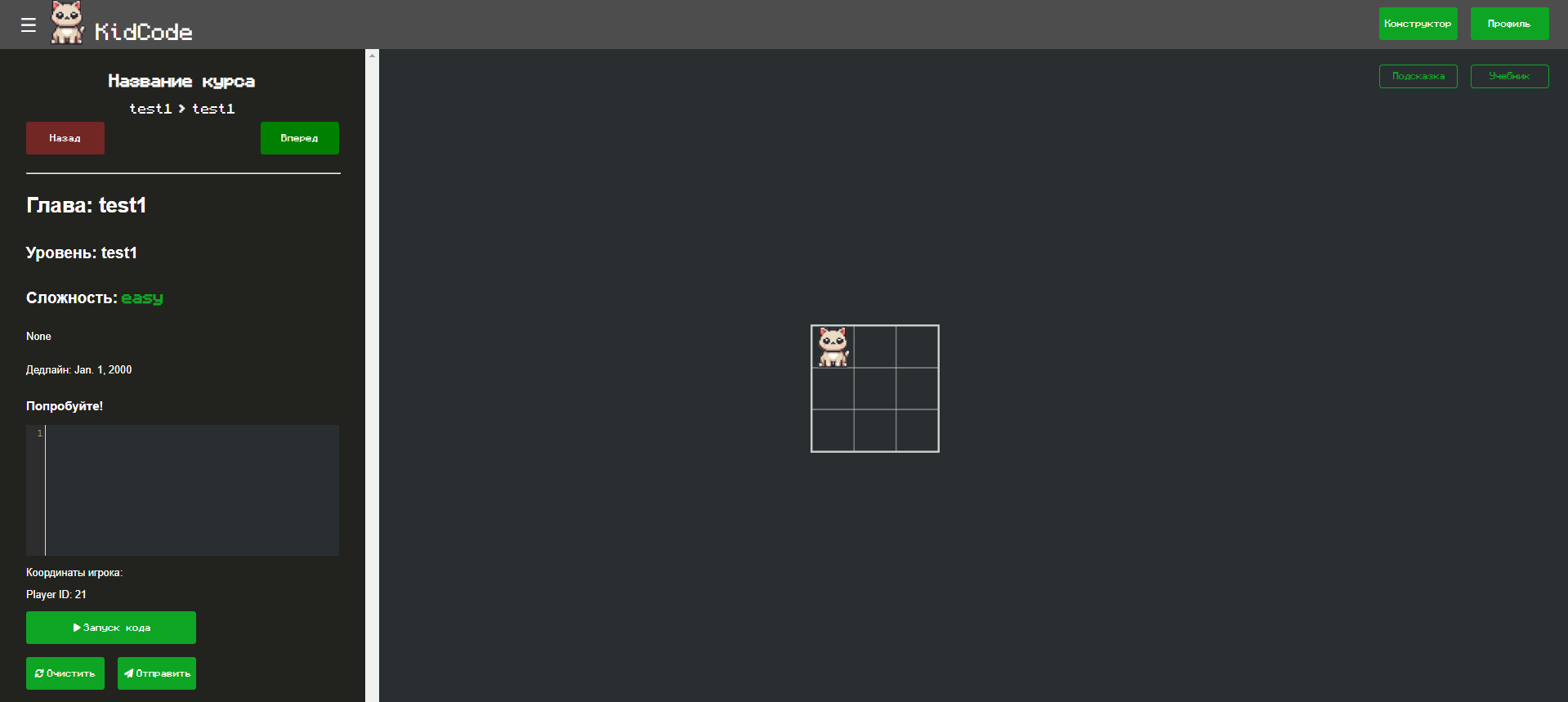


Рисунок 32 – Главная страница

На рисунке 33 приведена страница авторизации. На этой странице пользователь может пройти авторизацию, введя свои учётные данные: логин и пароль (для ученика) или пароль (для учителя), и нажать на кнопку «Войти». В случае не верно введенных данных пользователю выведется сообщение об этом, после чего он может ввести данные повторно.

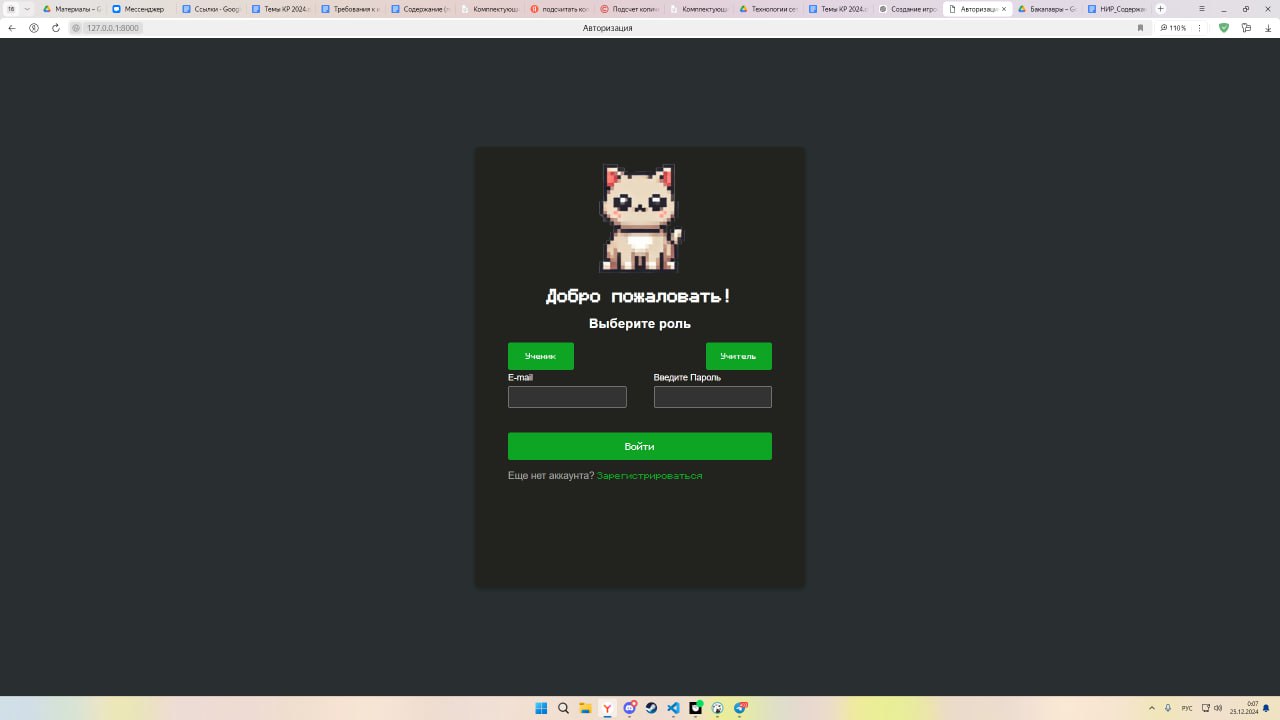
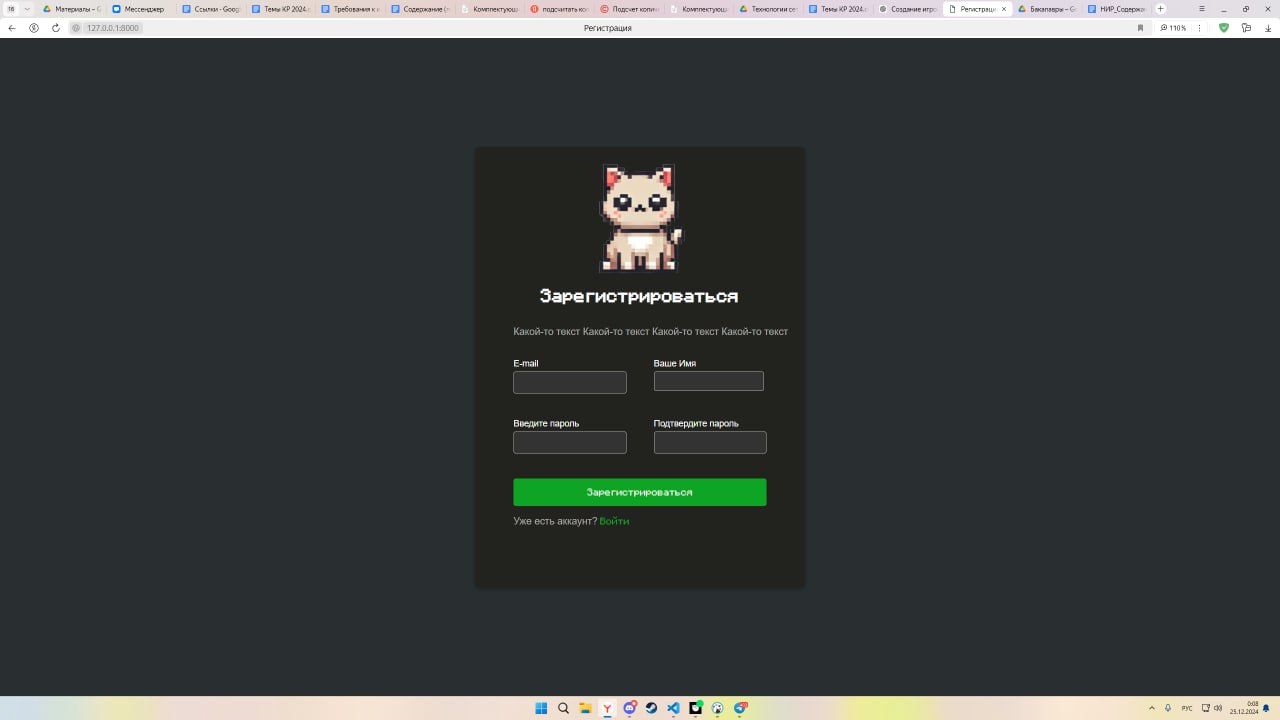


Рисунок 33 – Страница авторизации

Если пользователь ещё не был зарегистрирован, на странице предусмотрена гиперссылка «Зарегистрироваться», нажав на которую, будет открываться страница регистрации, вид которой приведена на рисунке 34. Здесь пользователь может создать новый аккаунт в системе, выбрав роль и введя данные, и нажать кнопку «Зарегистрироваться». При корректности данных пользователю выведется сообщение об удачной регистрации. В случае некорректности данных пользователь может ввести их повторно.

  
Рисунок 34 – Страница регистрации

Пользователь может перейти на страницу личного профиля, которая приведена на рисунке 35, нажав кнопку «Профиль». В нем он может просмотреть информацию о своих данных, отредактировать свое имя и пароль.

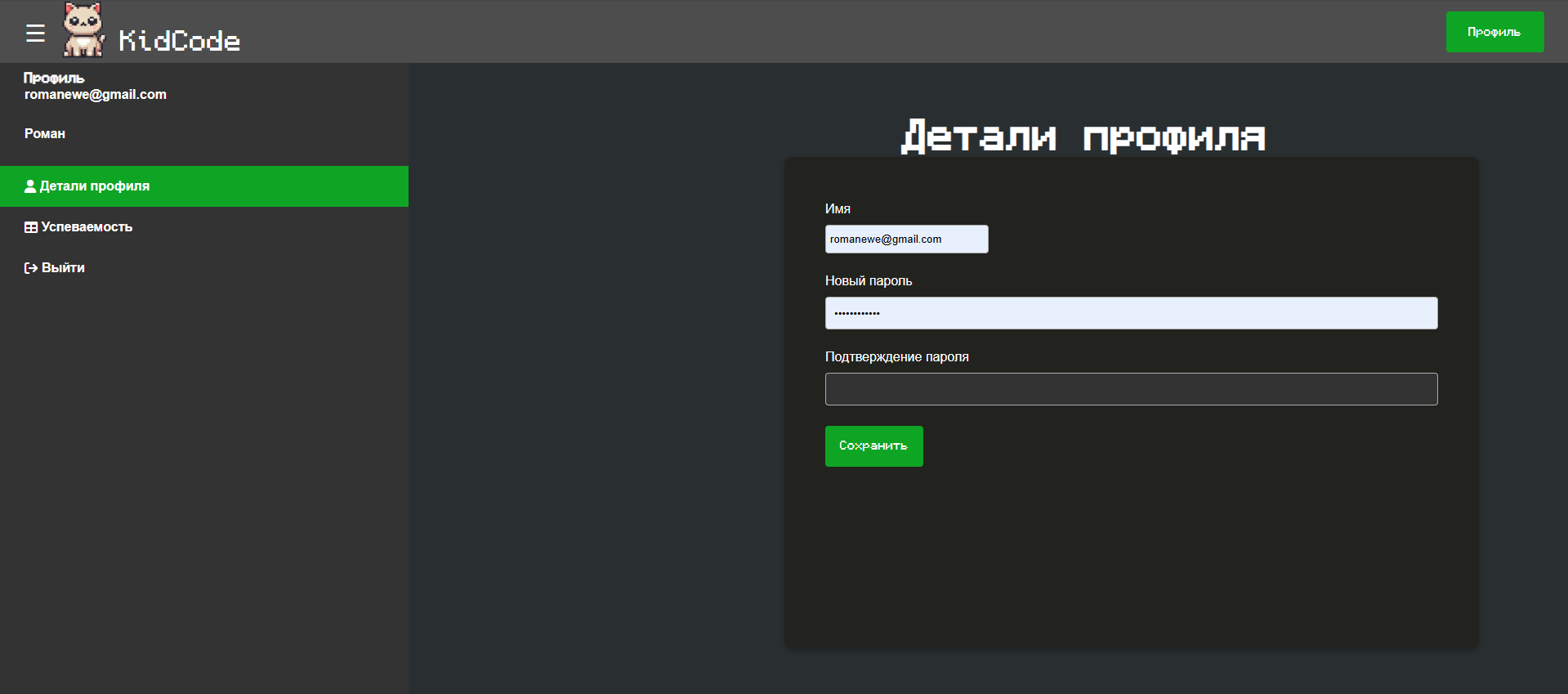


Рисунок 35 – Страница профиля

Для изменения пароля требуется ввести его повторно во второе поле для подтверждения. Также пользователь может перейти в раздел «Успеваемость» и просмотреть таблицу с оценками и выполненными заданиями, который представлен на рисунке 36. Применяя фильтры, пользователь может искать интересующую его информацию. Помимо этого, пользователь может выйти из аккаунта, нажав кнопку «Выйти».

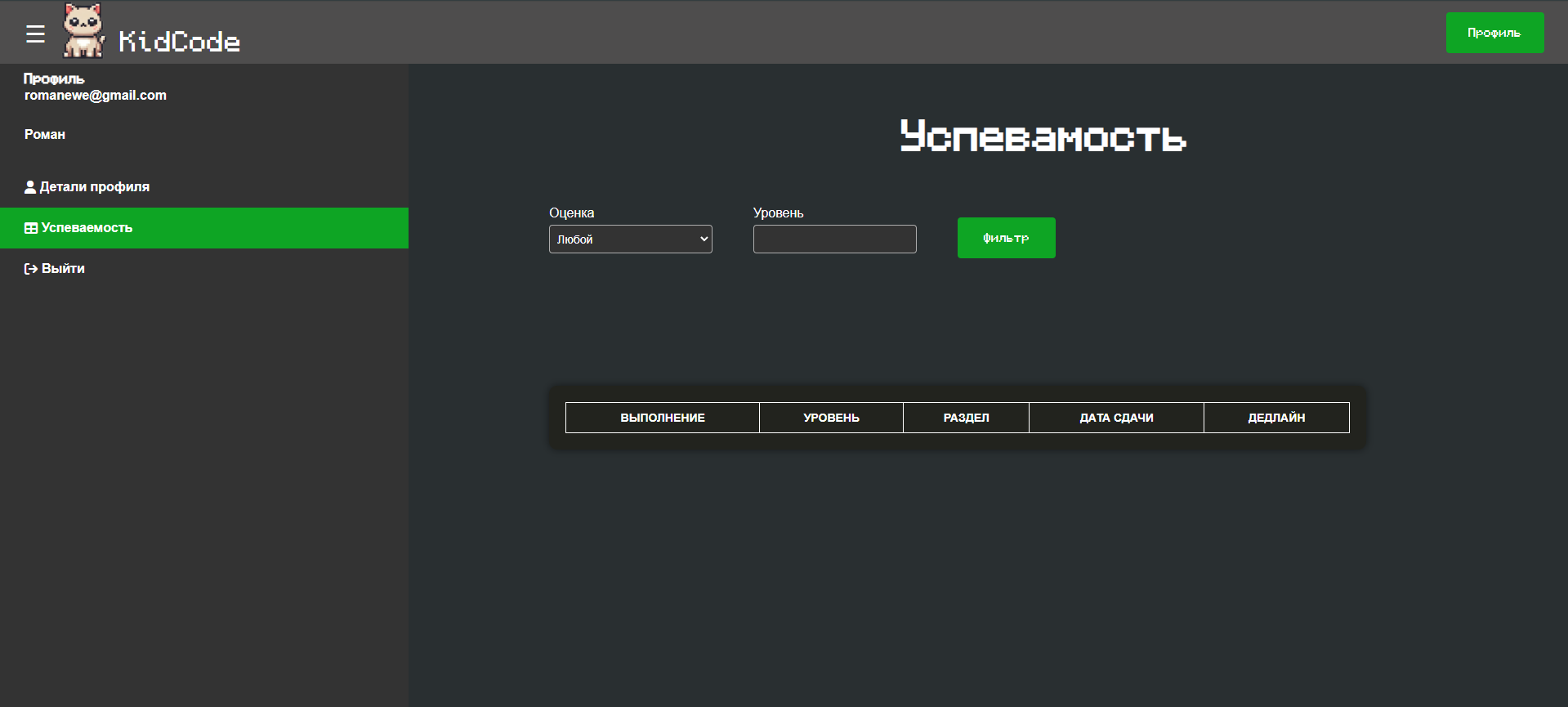


Рисунок 36 – Страница успеваемости

Если был выполнен вход в роли учителя, переход на страницу личного профиля и открыт раздел «Журнал», который представлен на рисунке 37, отобразится страница успеваемости учеников. Здесь учитель может применить фильтры для отображения интересующей его информации по полям «Статус», «Уровень» и «Имя».

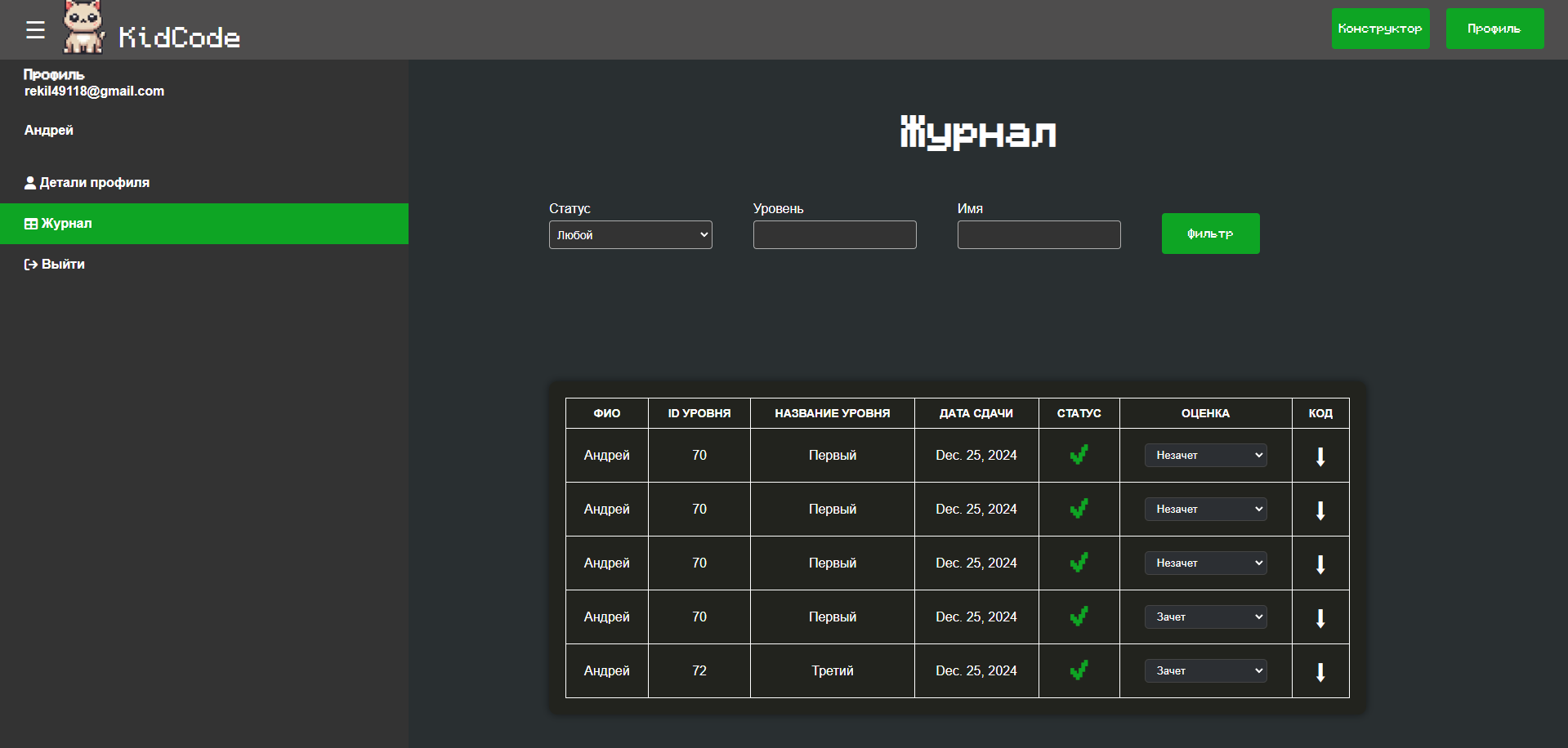


Рисунок 37 – Страница журнала

На рисунке 38 приведена страница конструктора. Здесь учитель создает задания для учеников. Учителю будет предложено два варианта создания задания: автоматически или вручную. При нажатии на кнопку «Автоматически» для генерации игрового поля учитель должен задать уровень сложности задания. Далее ввести текст задания и нажать кнопку «Сгенерировать». Если генерация прошла успешно, требуется нажать на кнопку «Сохранить». В случае, если поля введены некорректно, то учитель должен нажать кнопку «Очистить» для удаления данных в полях.

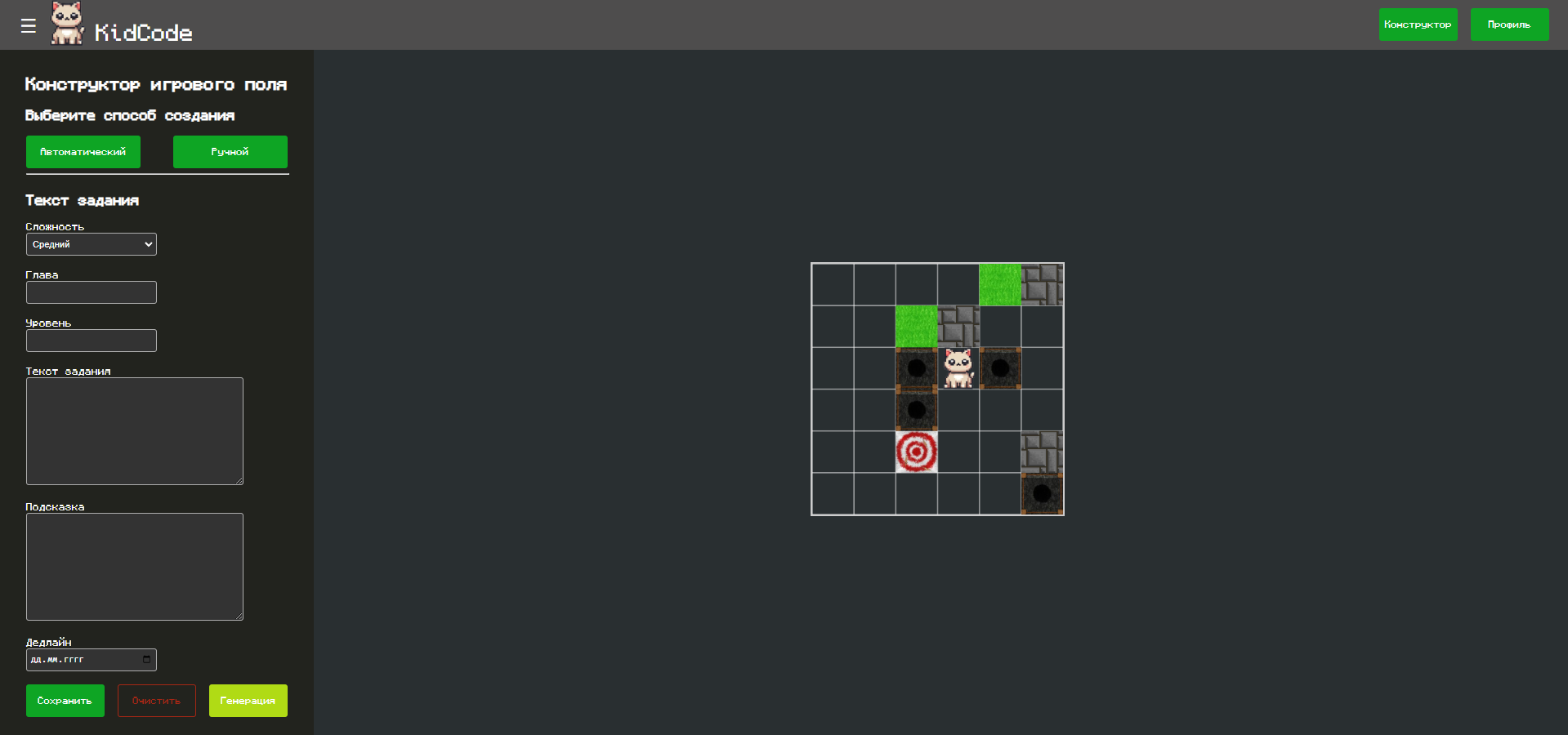


Рисунок 37 – Страница конструктора в автоматическом режиме

На рисунке 38 представлена страница создания задания в ручном режиме. По нажатию на кнопку «Вручную», учителю требуется ввести параметры игрового поля, выбрать шаблоны, выбрать уровень сложности и ввести текст задания. После заполнения полей, учителю требуется нажать кнопку «Сохранить». Если задание создано успешно, то задание сохраняется в БД. В случае, если поля введены некорректно, то учитель должен нажать кнопку «Очистить» для удаления данных в полях.

Страница выбора задания приведена на рисунке 39. По нажатию на выпадающий список в левом верхнем углу, пользователю будет представлен список глав, из которого он может выбрать нужную ему. После выбора главы пользователю будет представлен список заданий, который содержится в выбранной главе. Кликнув по одному из заданий, система откроет его содержимое.

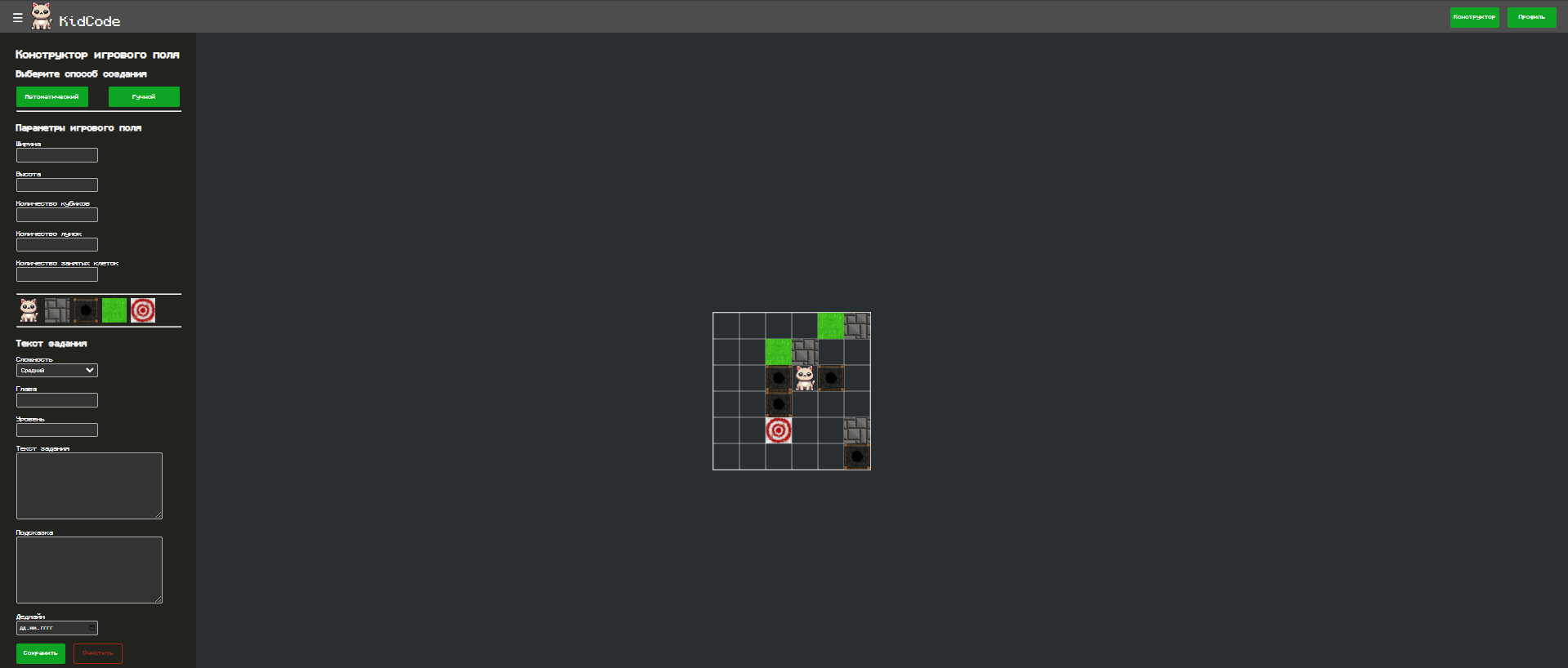


Рисунок 38 – Страница конструктора в ручном режиме

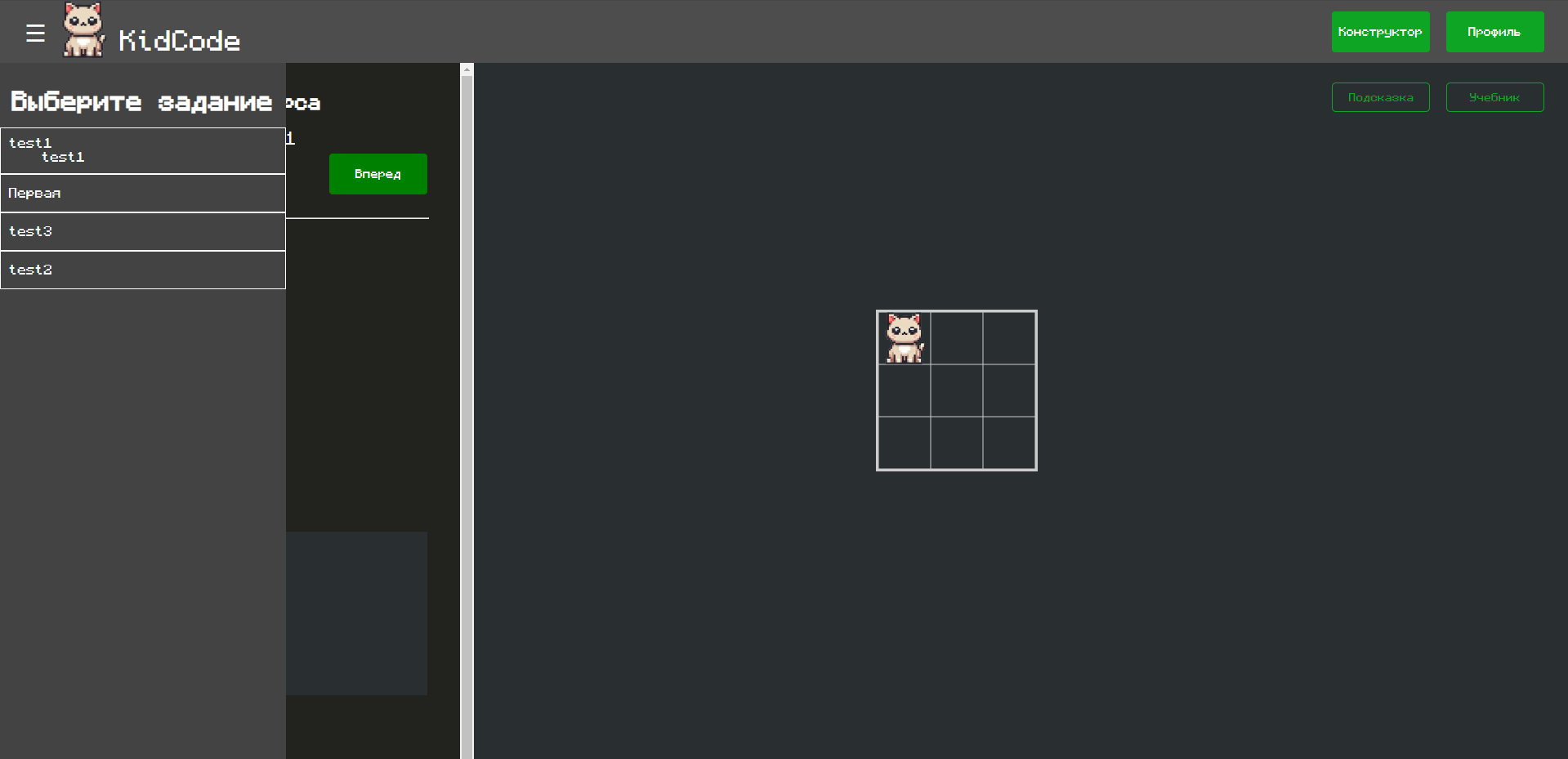


Рисунок 39 – Страница выбора задания

Страница задания представлена на рисунке 40. Здесь ученик может написать код в соответствии с заданием. После написания кода он должен нажать кнопку «Выполнить» или очистить поле написания кода, нажав на кнопку «Очистить». После успешного выполнения задания ученику требуется нажать кнопку «Отправить» для отправки кода учителю на проверку.

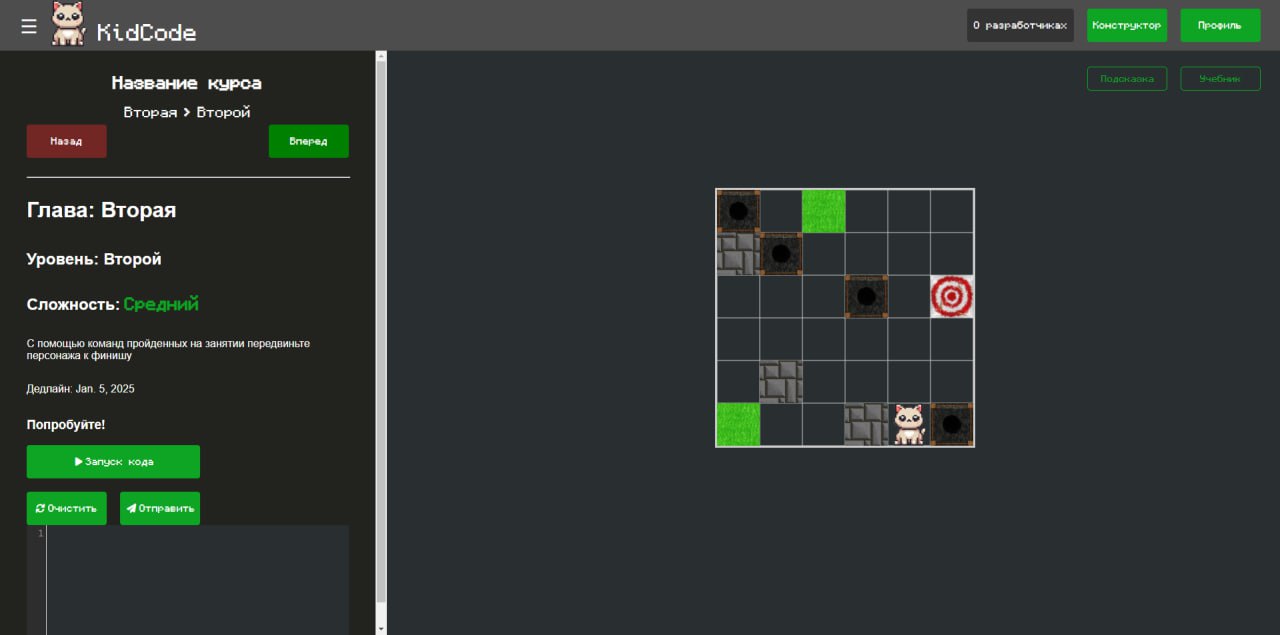


Рисунок 40 – Страница задания

Пользователь может посмотреть страницу «Справочная информация», нажав на соответствующую кнопку. На этой странице рассказывается о функциональных возможностях сайта, доступных пользователям, и описывает все ключевые функции и удобства, предлагаемые для обучения через веб-сервис. Вид страницы данного раздела приведён на рисунке 41.

На рисунке 42 приведено модальное окно «О разработчиках». Здесь пользователь может посмотреть сведения о команде, создавшей данное приложение.

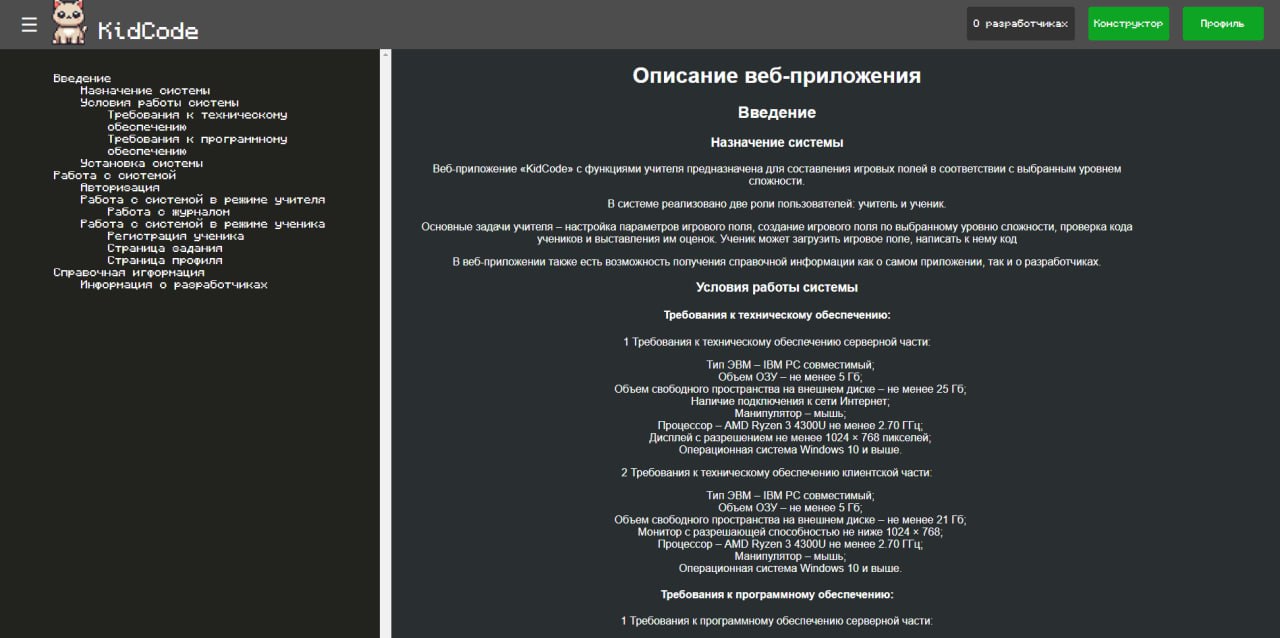


Рисунок 41 – Страница справочной информации

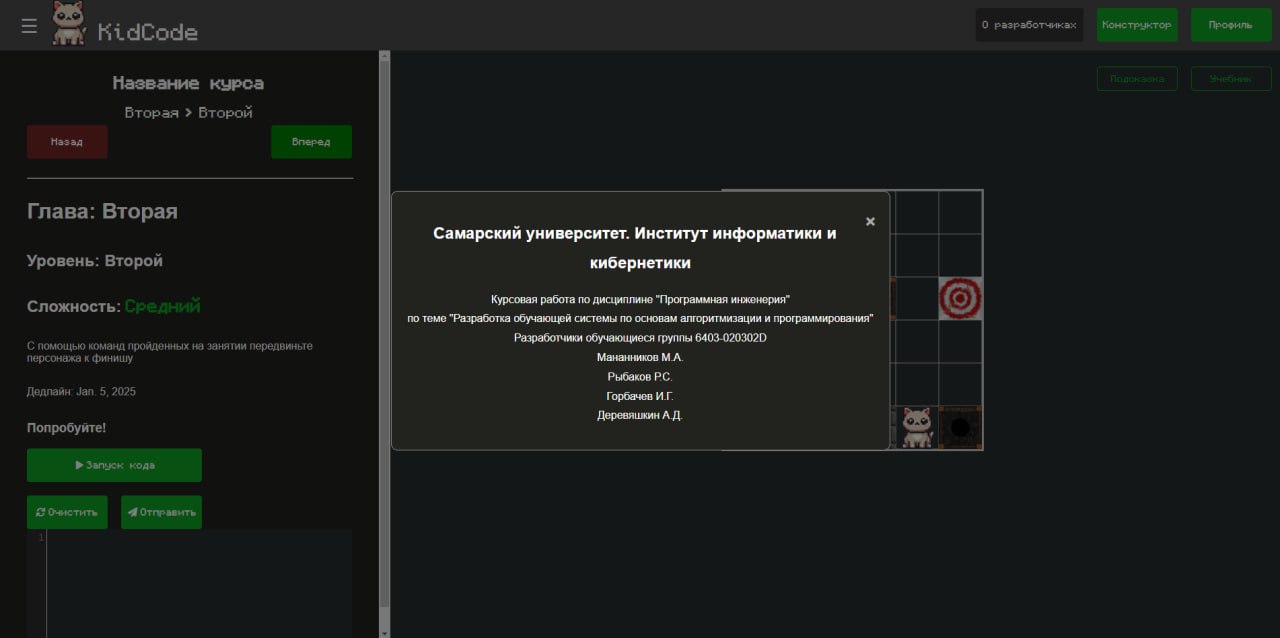


Рисунок 42 – Модальное окно с информацией о разработчиках

* 1. Диаграммы реализации

Диаграммы реализации предназначены для отображения состава компилируемых и выполняемых модулей системы, а также связей между ними. Диаграммы реализации разделяются на два конкретных вида: диаграммы компонентов (component diagrams) и диаграммы развертывания (deployment diagrams) [26].

* + 1. Диаграмма компонентов

Диаграмма компонентов описывает особенности физического представления системы [27].

На рисунке 39 приведена диаграмма компонентов, их описание приведено в таблице 10.

Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу.

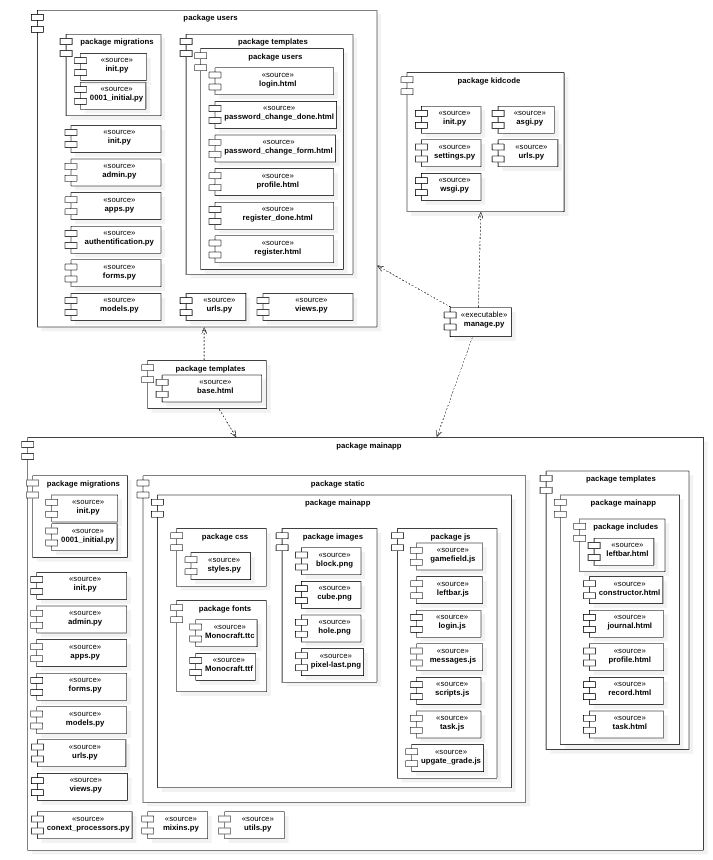


Рисунок 39 – Диаграмма компонентов системы

Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Таблица 10 – Описание компонентов системы

|  |  |  |
| --- | --- | --- |
| Название компонента | Назначение компонента | Подсистема |
| 1 | 2 | 3 |
| Manage.py | Запуск приложения и управление жизненным циклом | Серверная часть |
| Init.py | Инициализация приложения | Подсистема взаимодействия с клиентской частью |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 0001\_initial.py | Инициализация миграции | Подсистема взаимодействия с БД |
| Admin.py | Регистрация моделей БД в админ-панели | Подсистема взаимодействия с БД |
| Apps.py | Регистрация приложения | Подсистема взаимодействия с БД |
| Authentification.py | Специальная логика аутентификации пользователя | Подсистема аутентификации |
| Forms.py | Хранение форм для работы с данными | Подсистема взаимодействия с БД |
| Models.py | Хранение сущностей базы данных | Подсистема взаимодействия с БД |
| Urls.py | Хранения URL-адресов | Подсистема взаимодействия с клиентской частью |
| Views.py | Определение функций или классов, которые обрабатывают HTTP-запросы и возвращают HTTP-ответы | Подсистема взаимодействия с БД |
| Settings.py | Хранение значения пресетов приложения | Подсистема взаимодействия с клиентской частью |
| Wsgi.py | Обеспечение подключения к рабочим веб-серверам | Подсистема взаимодействия с БД |
| Asgi.py | Реализация асинхронного интерфейса | Подсистема взаимодействия с БД |
| Context\_processors.py | Добавление данных в контекст шаблонов | Подсистема взаимодействия с клиентской частью |
| Mixins.py | Создает миксины — классы, которые предоставляют определённый функционал, который может быть использован в других классах | Подсистема взаимодействия c БД |
| Utils.py | Организация вспомогательных функций и утилит | Подсистема взаимодействия с клиентской частью |
| Styles.css | Задание внешних стилей для html-разметки | Подсистема визуализации |
| Login.html | Разметка страницы авторизации | Подсистема регистрации и авторизации |
| Password\_change\_done.html | Разметка страницы успешной смены пароля | Подсистема регистрации и авторизации |
| Password\_change\_form.html | Разметка страницы смены пароля | Подсистема регистрации и авторизации |
| 1 | 2 | 3 |
| Profile.html | Разметка страницы профиля пользователя | Подсистема ввода данных |
| Register\_done.html | Разметка страницы успешной регистрации | Подсистема регистрации и авторизации |
| Register.html | Разметка страницы регистрации | Подсистема регистрации и авторизации |
| Leftbar.html | Разметка страницы левого навигационного меню | Подсистема визуализации |
| Constructor.html | Разметка страницы конструктора | Подсистема работы с заданием |
| Journal.html | Разметка страницы журнала | Подсистема работы с журналом |
| Record.html | Разметка страницы успеваемость | Подсистема работы с учеником |
| Task.html | Разметка страницы задания | Подсистема составления задания |
| Base.html | Базовый шаблон, от которого наследуются остальные HTML-страницы | Подсистема визуализации |
| Gamefield.js | Сценарий генерации игрового поля | Подсистема работы с игровым полем |
| Leftbar.js | Сценарий выдвижения навигационного меню | Подсистема визуализации |
| Login.js | Сценарий смены роли пользователя при авторизации | Подсистема авторизации и регистрации |
| Messages.js | Сценарий показа/закрытия всплывающих окон | Подсистема визуализации |
| Scripts.js | Сценарий осуществления перехода на страницу задания | Подсистема взаимодействия с серверной частью |
| Task.js | Сценарий работы с заданием | Подсистема взаимодействия с серверной частью |
| Upgate\_grade.js | Сценарий смены оценки | Подсистема взаимодействия с серверной частью |

* + 1. Диаграмма развертывания

Диаграмма развертывания предназначена для представления общей конфигурации или топологии распределенной программной системы и содержит изображение размещения различных артефактов по отдельным узлам системы [28]. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

На рисунке 40 приведена диаграмма развертывания системы.

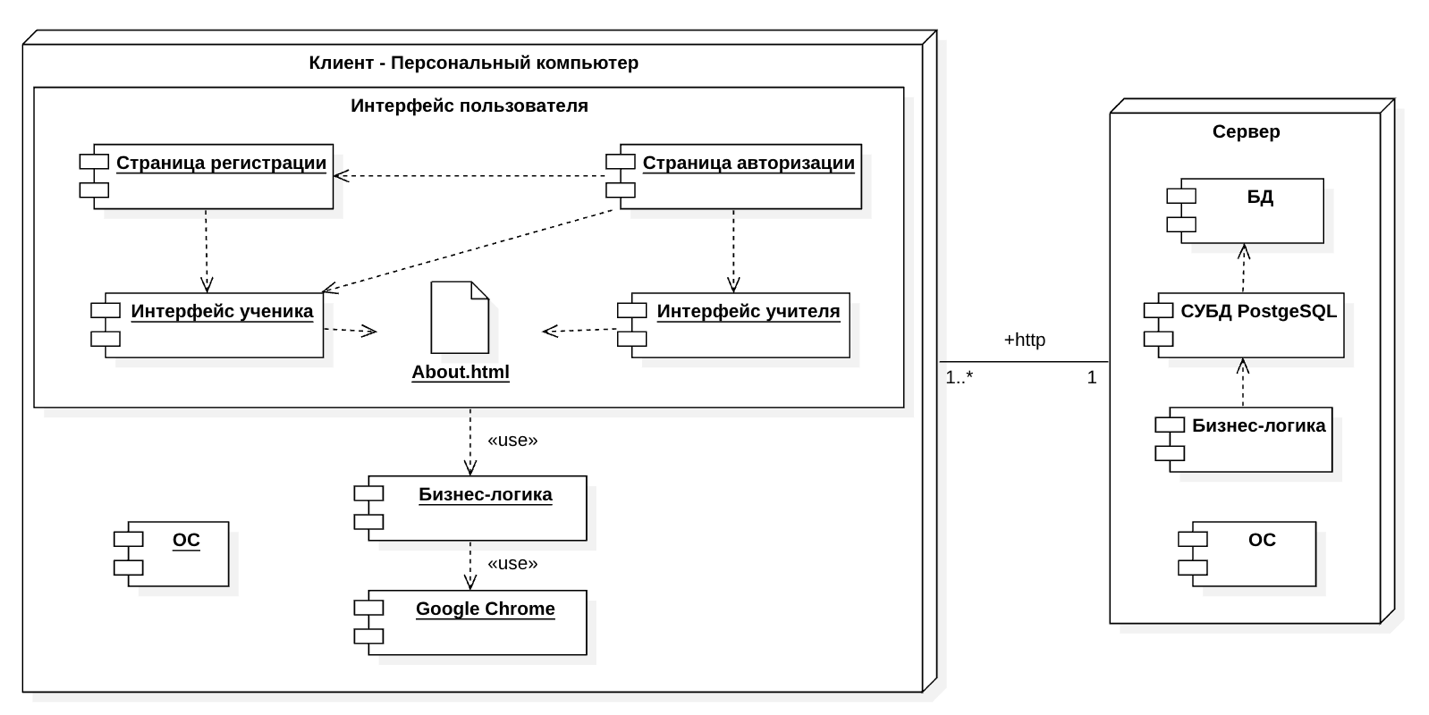


Рисунок 40 – Диаграмма развертывания системы

* + 1. Диаграмма классов

В соответствии со спецификацией, приведенной в п. 2.5.6, и с учетом выбранного языка программирования (см. п. 2.7.1) разработана диаграмма классов системы (этап реализации), приведенная на рисунке 41.

* + 1. Физическая модель базы данных

Физическое проектирование является последним этапом проектирования базы данных, при выполнении которого принимается решение о способах реализации разрабатываемой базы данных. Во время логического проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области).

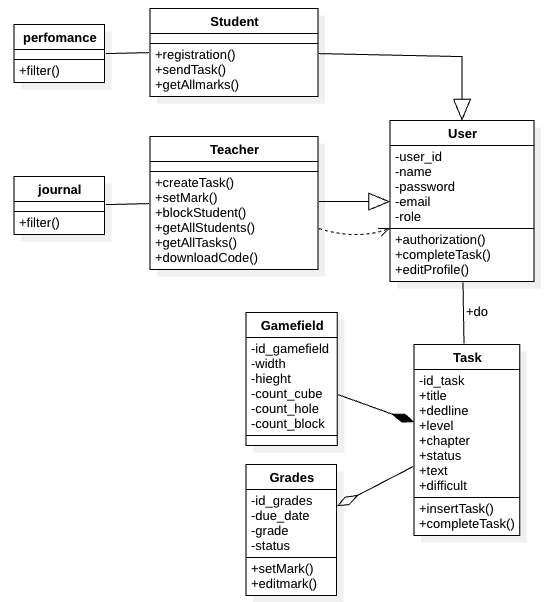


Рисунок 41 – Описание классов системы (этап реализации)

Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей [29].

На рисунке 42 представлена физическая модель данных системы.

В таблицах 11-15 приведено описание сущностей БД. Первичные ключи выделены жирным шрифтом, а внешние – курсивом.

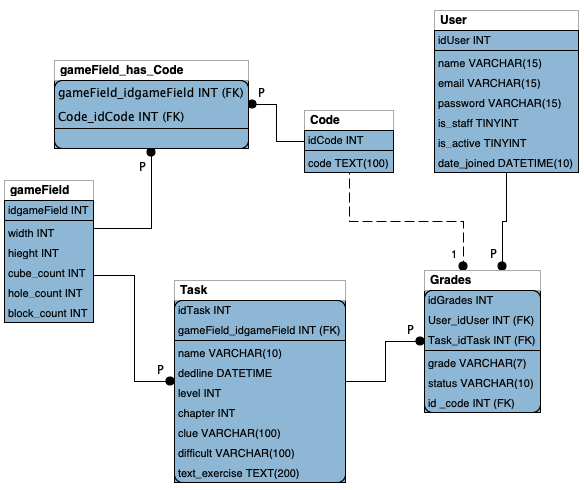
  
Рисунок 42 – Физическая модель данных системы

Таблица 11 – Сущность «User»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **User Id** | **uniqueidentifier** | **int** | **8** |
| Name | Имя пользователя | varchar(15) | 15 |
| Password | Пароль | varchar(15) | 15 |
| e-mail | Адрес электронной почты | varchar(15) | 15 |
| is\_staff | Учитель или ученик | tinyint | 1 |
| is\_active | Активный или заблокированный | tinyint | 1 |
| date\_joined | Дата добавления | datetime(10) | 4 |
| Размер записи | | | 59 |

Таблица 12 – Сущность «Grades»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| 1 | 2 | 3 | 4 |
| **Grades Id** | **uniqueidentifier** | **int** | **8** |
| *user Id* | *Идентификатор пользователя* | *int* | *8* |
| status | Статус выполнения | varchar(10) | 10 |
| 1 | 2 | 3 | 4 |
| grade | Оценка | varchar(7) | 7 |
| *task id* | *Идентификатор задания* | *int* | *8* |
| *code id* | *Идентификатор кода* | *int* | *8* |
| Размер записи | | | 49 |

Таблица 13 – Сущность «Task»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **Task Id** | **uniqueidentifier** | **int** | **8** |
| name | Название уровня | varchar(10) | 10 |
| dedline | Срок сдачи | datetime | 4 |
| level | Уровень | int | 8 |
| chapter | Глава | int | 8 |
| clue | Текст подсказки | varchar(100) | 100 |
| difficult | Сложность задания | enum | 3 |
| text\_exercise | Текст задания | varchar(200) | 200 |
| *game\_field\_id* | *Идентификатор игрового поля* | *int* | *8* |
| Размер записи | | | 349 |

Таблица 14 – Сущность «gameField»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| 1 | 2 | 3 | 4 |
| **gameFileld Id** | **uniqueidentifier** | **int** | **8** |
| width | Ширина поля | int | 8 |
| 1 | 2 | 3 | 4 |
| height | Высота поля | int | 8 |
| cube\_count | Количество кубиков | int | 8 |
| hole\_count | Количество лунок | int | 8 |
| block\_count | Количество блоков | int | 8 |
| Размер записи | | | 48 |

Таблица 15 – Сущность «Сode»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **Code Id** | **uniqueidentifier** | **int** | **8** |
| code | Текст кода | text | без ограничений |
| Размер записи | | | 8 |

* 1. Выбор и обоснование комплекса технических средств
     1. Расчет объема занимаемой памяти

Расчет объема внешней памяти серверной части

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования системы, воспользуемся следующей формулой:

VСЧ = VОС+ VPIP + VПР + VБД,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 10 Home 64 бит, VОС = 16 Гб);

VPIP – объем памяти, занимаемый PIP (дадим оценку сверху VPIP в 130 Мб);

VПР – объем памяти, занимаемый непосредственно файлами приложения (VПР = 60 Мб);

VБД – объем памяти, занимаемый базой данных (всеми таблицами) при ее максимальном заполнении. Расчет этой составляющей приведен в таблице 19 (VБД = 31050 байт ~ 0,03 Мб).

Таким образом, суммарный объем внешней памяти составит:

VСЧ = 16 Гб + 130 Мб + 60 Мб + 0,03 Мб ~ 16,2 Гб.

Таблица 16 – Расчет объема внешней памяти, необходимой для хранения БД

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Размер записи (байт) | Максимум записей | Всего (байт) |
| Пользователь | 59 | 100 | 5900 |
| Код | 8 | 50 | 400 |
| Оценки | 49 | 100 | 4900 |
| Задание | 349 | 50 | 17450 |
| Игровое поле | 48 | 50 | 2400 |
| Итого | | | 31050 |

Расчет объема внешней памяти клиентской части

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования системы, воспользуемся следующей формулой:

VКЧ = VОС+ VБР,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 10 Home 64 бит, VОС = 16 Гб);

VБР – объем памяти, занимаемый браузером Google Chrome 96.0.4664.110 (дадим оценку сверху VБР в 800 Мб).

Таким образом, суммарный объем внешней памяти составит:

VСЧ = 16 Гб + 800 Мб ~ 17 Гб.

Расчет объема ОЗУ серверной части

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ\_СЧ = VОС+ VPIP + VПР + VБД,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

VPIP – ОЗУ, занимаемое PIP (не превысит 200 Мб);

VПР – ОЗУ, которое займет само приложение (дадим ему оценку сверху в 50 Мб);

VБД – объем данных из базы, который может быть одновременно загружен в оперативную память (дадим ему оценку сверху в 1 Мб).

Суммарные объемы ОЗУ составит:

VОЗУ\_СЧ = 2 Гб + 200 Мб + 50 МБ + 1 Мб ~ 2.3 Гб.

Таким образом, 2.3 Гб оперативной памяти можно счесть минимально необходимым для функционирования серверной части системы.

Расчет объема ОЗУ клиентской части

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ\_КЧ = VОС+ VБР,

где VОС – ОЗУ, занимаемое операционной системой (2 Гб);

VБР – объем данных из базы, который может быть одновременно загружен в оперативную память (дадим ему оценку сверху в 100 Мб).

Суммарные объемы ОЗУ составит:

VОЗУ\_КЧ = 2 Гб + 100 Мб ~ 2.1 Гб.

Таким образом, 2.1 Гб оперативной памяти можно счесть минимально необходимым для функционирования клиентской части системы.

* + 1. Минимальные требования, предъявляемые к системе

Для корректного функционирования клиентской части системы необходимо:

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 2 Гб;
* объем свободного дискового пространства – не менее 16,2 Гб;
* клавиатура или иное устройство ввода;
* мышь или иное манипулирующее устройство;
* процессор – Intel Pentium не менее 1,5 ГГц;
* дисплей с разрешением не менее 1024 × 768 пикселей;
* операционная система Windows 7 и выше;
* подключение к интернету;
* браузер Google Chrome 86.0.4240.183 (64-битный) и выше, Firefox 83.0 (64-битный) и выше.

Для корректного функционирования серверной части системы необходимо:

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 3 Гб;
* объем свободного дискового пространства – не менее 16,2 Гб;
* процессор – Intel Pentium не менее 1,5 ГГц;
* операционная система Windows 7 и выше;
* подключение к интернету;
* браузер Google Chrome 86.0.4240.183 (64-битный) и выше, Firefox 83.0 (64-битный) и выше.
* среда программирования – Visual Studio Code 1.96.2 и выше;
* СУБД – PostgreSQL 17.0 и выше.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта было разработано веб-приложение, для обучения школьников основам алгоритмизации и программирования. В нем пользователи могут выполнять задания путем написания кода, а учитель следить за их выполнением.

В первом разделе описана предметная область и приведены основные её понятия, рассмотрены и проанализированы характеристики систем-аналогов, приведён их сравнительный анализ. На основе проведённого анализа выполнена объектная декомпозиция, отражённая в диаграмме объектов, и сформулирована постановка задачи.

Во втором разделе было реализовано проектирование системы. Была выбрана архитектура приложения, разработана структурная схема, определена функциональная спецификация и разработан информационно-логический проект системы с помощью методологии UML. Также были разработаны прототипы страниц приложения, логическая модель БД и выбраны программные средства для его реализации.

В третьем разделе была описана реализация системы. Был описан интерфейс пользователя, представлены диаграммы реализации и описана физическая модель данных. Были сделаны ресурсные расчёты и определены технические требования для функционирования системы.

Разработанная система может использоваться образовательными учреждениями для организации процесса обучения в формате онлайн.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Изд. 2-е. М.: ДМК Пресс, 2006. С. 21.

Буч Г., Рамбо Д., Якобсон А. Язык UML С. 31.

Предметная область (Object domain) // Вики [Электронный ресурс]. URL: https://wiki.loginom.ru/articles/object-domain.html (дата обращения: 18.09.2024).

Объектно-ориентированное проектирование // Википедия. [2022]. [Электронный ресурс]. URL: [https://ru.wikipedia.org/?curid=1115148&oldid  
=119347606](https://ru.wikipedia.org/?curid=1115148&oldid=119347606) (дата обращения: 19.09.2024).

Методология объектно-ориентированного анализа и проектирования // Studfile URL: https://studfile.net/preview/2081602/page:2/ (дата обращения: 19.09.2024).

Система [Электронный ресурс]. URL: https://www.booksite.ru/  
fulltext/1/001/008/102/619.htm (дата обращения: 02.10.2024).

Проектирование [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/проектирование (дата обращения: 02.10.2024).

Определение термина "Архитектура" // Майкрософт [Электронный ресурс]. URL: https://microsoft.fandom.com/ru/wiki/архитектура (дата обращения: 03.10.2024).

Дергачев А.М., Кореньков Ю.Д., Логинов И.П., Сафронов А.Г., Технологии веб-сервисов. СПб: Университет ИТМО, 2021. 100 с.

Клиент-серверная архитектура [Электронный ресурс]. URL: https://pikabu.ru/story/karera\_v\_it\_sistemnyiy\_analitik\_chast\_6\_klientservernaya\_arkhitektura\_10206499 (дата обращения: 03.10.2024).

Разъясняем: что такое веб-приложение и зачем оно нужно? [Электронный ресурс]. URL: https://sibdev.pro/blog/articles/chto-takoe-veb-prilozhenie(дата обращения: 03.10.2024).

Структурный подход к проектированию программного обеспечения [Электронный ресурс]. URL: https://studfile.net/preview/1966781/  
page:2/ (дата обращения: 03.10.2024).

Как написать спецификацию требований к программному обеспечению [Электронный ресурс]. URL: https://stfalcon.com/ru/blog/post/  
How-to-Write-a-Software-Requirements-Specification (дата обращения: 14.10.2024).

Что такое интерфейс? [Электронный ресурс]. URL: https://www.uprock.ru/education/chto-takoe-interfeys (дата обращения: 14.10.2024).

Зачем нужно прототипирование интерфейсов? [Электронный ресурс]. URL: https://webdoka.ru/blog/zachem-nuzhno-prototipirovanie-ux/ (дата обращения: 30.10.2024).

Что такое UML? [Электронный ресурс]. URL: https://habr.com/ru/articles/458680/ (дата обращения: 31.10.2024).

Лекция 4: Спецификация требований и рекомендации по написанию эффективных вариантов использования // Национальный Открытый Университет «ИНТУИТ» URL: https://intuit.ru/studies/courses/32/32/lecture/4. (дата обращения: 01.11.2024).

Диаграмма классов [Электронный ресурс] URL: https://studopedia.info/10-59449.html (дата обращения: 27.11.2024).

Диаграмма состояний (statechart diagram) // URL: http://imlearning.ru/netcat\_files/file/FSIS/диаграмма\_состояний.pdf (дата обращения: 14.11.2024).

1.3 Диаграммы деятельности Диаграммы деятельности. // НГТУ НЭТИ URL: https://ciu.nstu.ru/kaf/persons/1914/page47048/diagramm\_  
deyatelnosti (дата обращения: 28.11.2024).

Диаграмма последовательности URL: https://studfile.net/preview/  
5187971/page:4/ (дата обращения: 28.11.2024).

Модель «Сущность-связь» [Электронный ресурс] URL: https://studopedia.ru/7\_172654\_model-sushchnost--svyaz.html (дата обращения: 24.11.2024).

Краткое описание ER-метода [Электронный ресурс] // Кафедра алгоритмических языков ВМК МГУ: [сайт]. URL: https://al.cs.msu.ru/system/files/ER\_method.pdf (дата обращения: 03.12.2024).

Введение в JavaScript [Электронный ресурс] URL: https://learn.javascript.ru/intro/ (дата обращения: 09.12.2024).

Основная информации о PostgreSQL [Электронный ресурс] URL: <https://www.opennet.ru/docs/RUS/mysqlcli/glava01.html> (дата обращения: 05.12.2024).

Диаграммы реализации [Электронный ресурс] // maksakov-sa.ru: [сайт]. URL: http://www.maksakov-sa.ru/ModelUML/DiagrReal/index.html (дата обращения: 23.12.2024).

1.1. Диаграмма компонентов (component diagram) [Электронный ресурс]. URL: http://imlearning.ru/netcat\_files/диаграмма\_компонентов.pdf (дата обращения: 09.12.2024).

1.1. Диаграмма развертывания (deployment diagram) // imlearning.ru [Электронный ресурс]. URL: http://imlearning.ru/netcat\_files/file/FSIS/  
диаграмма\_развертывания.pdf (дата обращения: 09.12.2024).

Физическое проектирование базы данных [Электронный ресурс] URL: https://studopedia.ru/8\_74489\_fizicheskoe-proektirovanie-bazi-dannih.html (дата обращения: 23.12.2024).

ПРИЛОЖЕНИЕ А  
Руководство пользователя

А.1 Назначение системы

Система предназначена для обучения программированию. Она предоставляет пользователям писать код, отправлять его на проверку учителю, просматривать личную информацию профиля и просматривать успеваемость. Учитель может проверять код, отправленный учеником, и выставлять оценку.

А.2 Условия работы системы

Для корректной работы клиентской части системы необходимо наличие соответствующих программных и аппаратных средств.

1. Требования к техническому обеспечению:

* ЭВМ типа: x86-64 совместимый;
* объем ОЗУ – не менее 3 ГБ;
* объем свободного дискового пространства – не менее 2 ГБ;
* клавиатура или иное устройство ввода;
* мышь или иное манипулирующее устройство;
* процессор – Intel Pentium не менее 1,5 ГГц;
* дисплей с разрешением не менее 1024 × 768 пикселей.

1. Требования к программному обеспечению:

* операционная система: Windows 7 и выше;
* подключение к интернету;
* браузер: Google Chrome 86.0.4240.183 (64-битный) и выше; Firefox 83.0 (64-битный) и выше.

Для корректной работы серверной части системы необходимо наличие соответствующих программных и аппаратных средств.

1. Требования к техническому обеспечению

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 8 ГБ;
* объем свободного дискового пространства – не менее 20 ГБ;
* процессор – Intel Core i5 или выше.

1. Требования к программному обеспечению

* операционная система: Windows 7 и выше;
* подключение к интернету;
* среда программирования: Visual Studio Code 1.9.2 и выше;
* СУБД: PostgreSQL 17.0 и выше.

А.3 Установка системы

Система KidCode поставляется в виде ZIP-архива с именем KidCode.zip. Данный файл необходимо распаковать в любую директорию на жестком диске. В результате распаковки создается папка KidCode, содержащая все необходимые файлы и каталоги для работы системы.

Система использует порт 8080 по умолчанию. Необходимо, чтобы этот порт был свободен и доступен.

Для установки системы необходимо перейти в папку kidcode, установить виртуальное окружение Python. Затем необходимо активировать виртуальное окружение и установить необходимые пакеты данных.

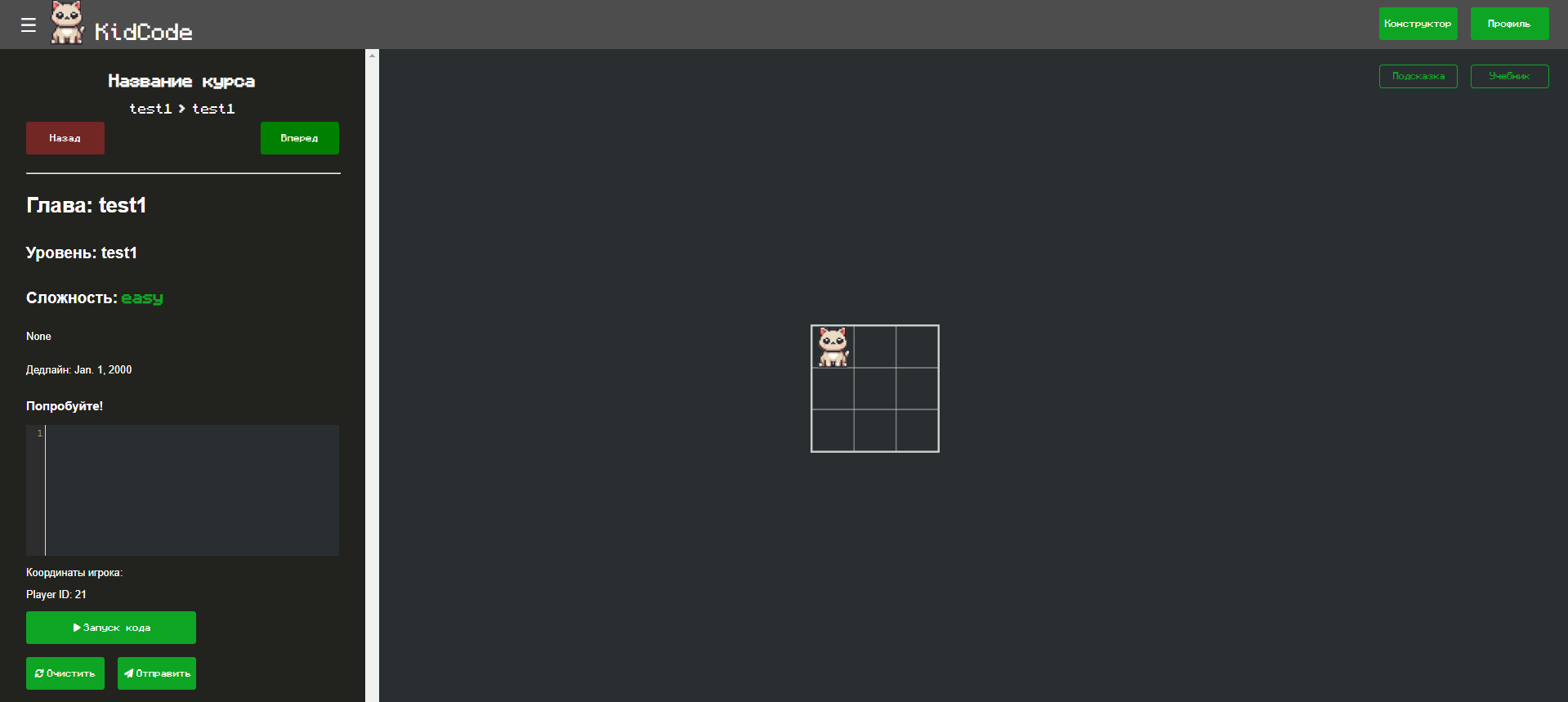
А.3.2 Запуск

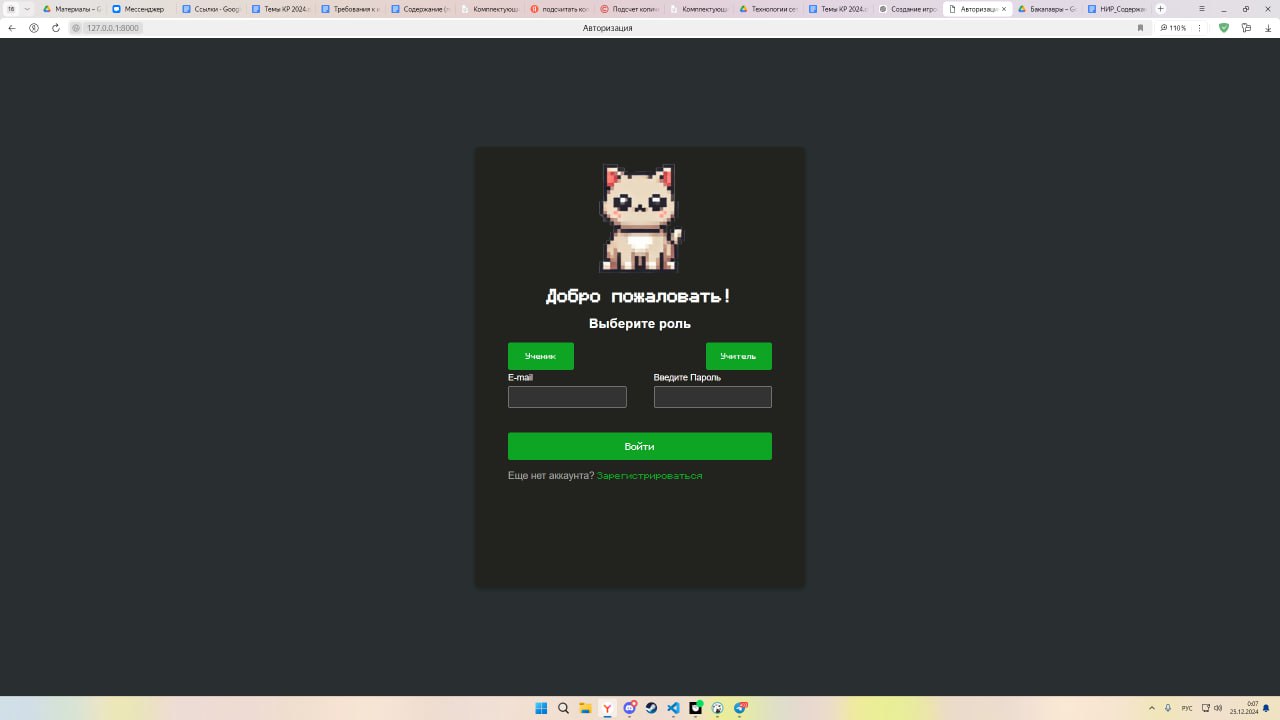
В проекте перейдите по следующему пути: kidcode/kidcode/. В этой папке откройте файл manage.py и запустите его с помощью Ctrl +F5. Сайт будет доступен по ссылке (локально) <http://localhost:8080/>.

А.4 Работа с системой

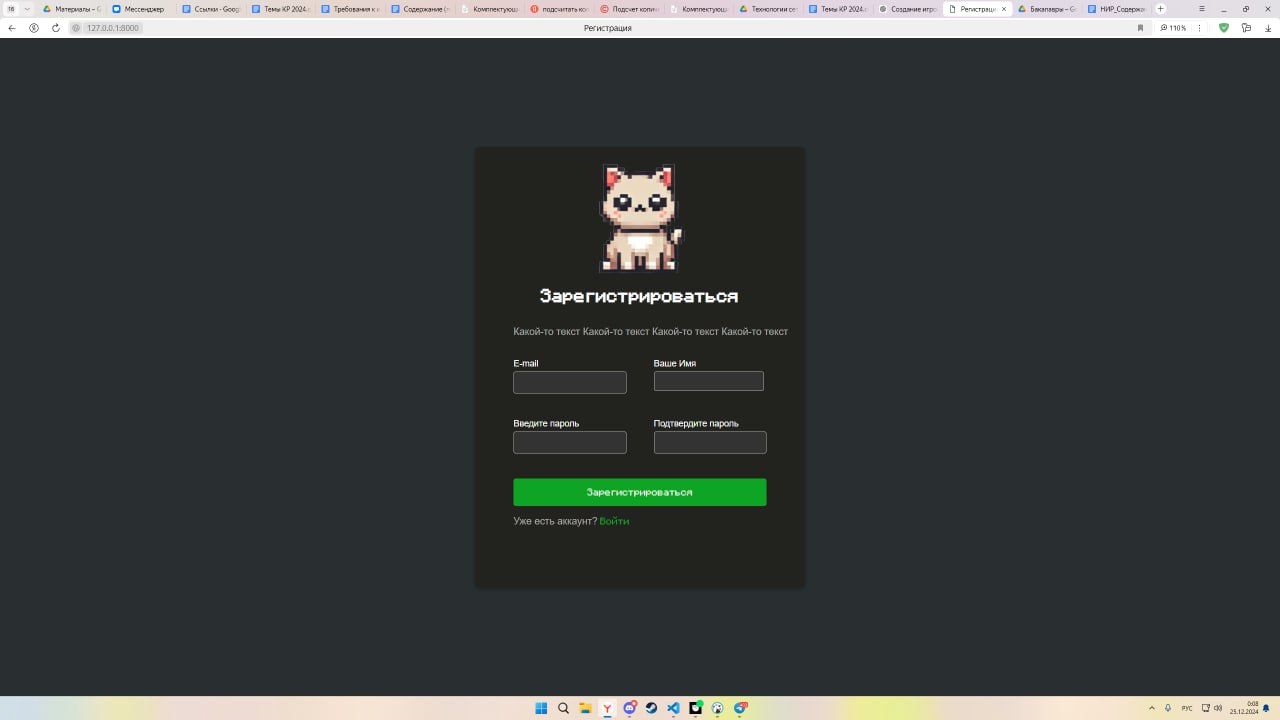
При запуске приложения открывается главная страница, которая приведена на рисунке А.1. В верхней части страницы на каждой странице расположено навигационное меню.

На рисунке А.2 приведена страница авторизации. На этой странице пользователь может пройти авторизацию, введя свои учётные данные: логин и пароль (для ученика) или пароль (для учителя), и нажать на кнопку «Войти». В случае не верно введенных данных пользователю выведется сообщение об этом, после чего он может ввести данные повторно.

  
Рисунок А.1 – Начальная страница

  
Рисунок А.2 – Страница авторизации

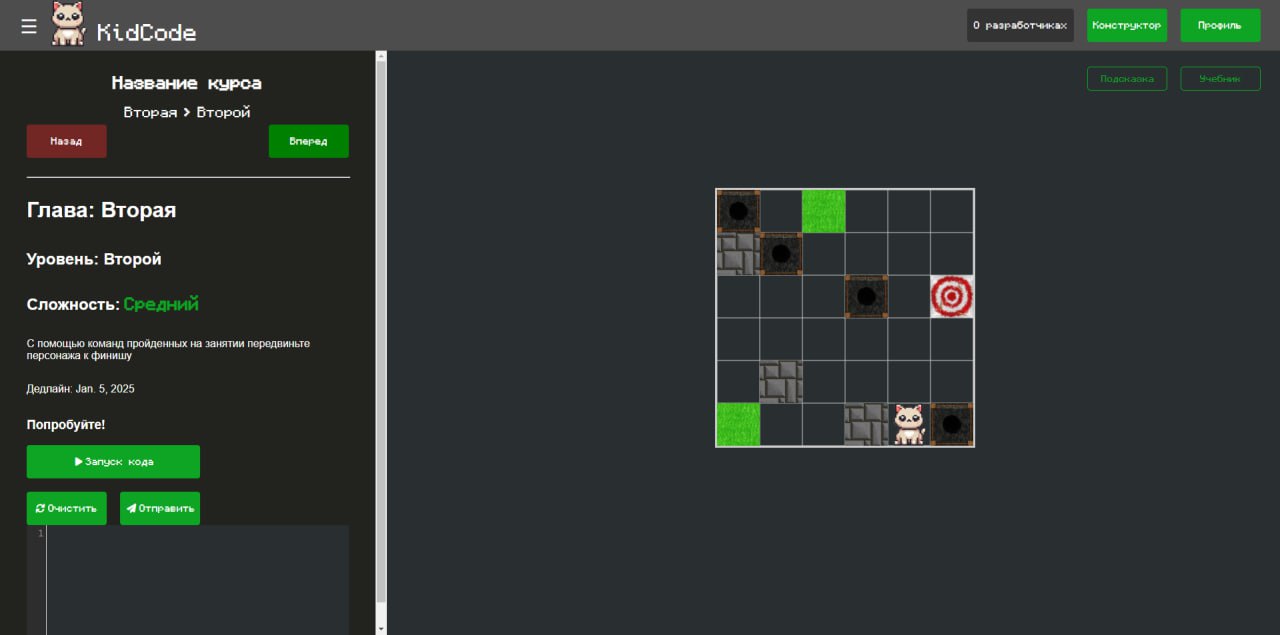
Если пользователь ещё не был зарегистрирован, на странице предусмотрена гиперссылка «Зарегистрироваться», нажав на которую, будет открываться страница регистрации, вид которой приведена на рисунке А.3. Здесь пользователь может создать новый аккаунт в системе, выбрав роль и введя данные, и нажать кнопку «Зарегистрироваться». При корректности данных пользователю выведется сообщение об удачной регистрации. В случае некорректности данных пользователь может ввести их повторно.

  
Рисунок А.3 – Страница регистрации

А.4.1 Работа с системой в режиме пользователя

Пользователь может просматривать задания, вид страницы которой приведён на рисунке А.4. На этой странице пользователь может написать код и запустить его на выполнение. Также пользователь может перейти в личный профиль по кнопке «Профиль».

На странице описания профиля пользователь может изменить личные данные и просмотреть успеваемость, если был выполнен вход в роли ученика или просмотреть журнал, если был выполнен вход в роли учителя. Вид страницы задания представлен на рисунке А.5

  
Рисунок А.4 – Страница задания

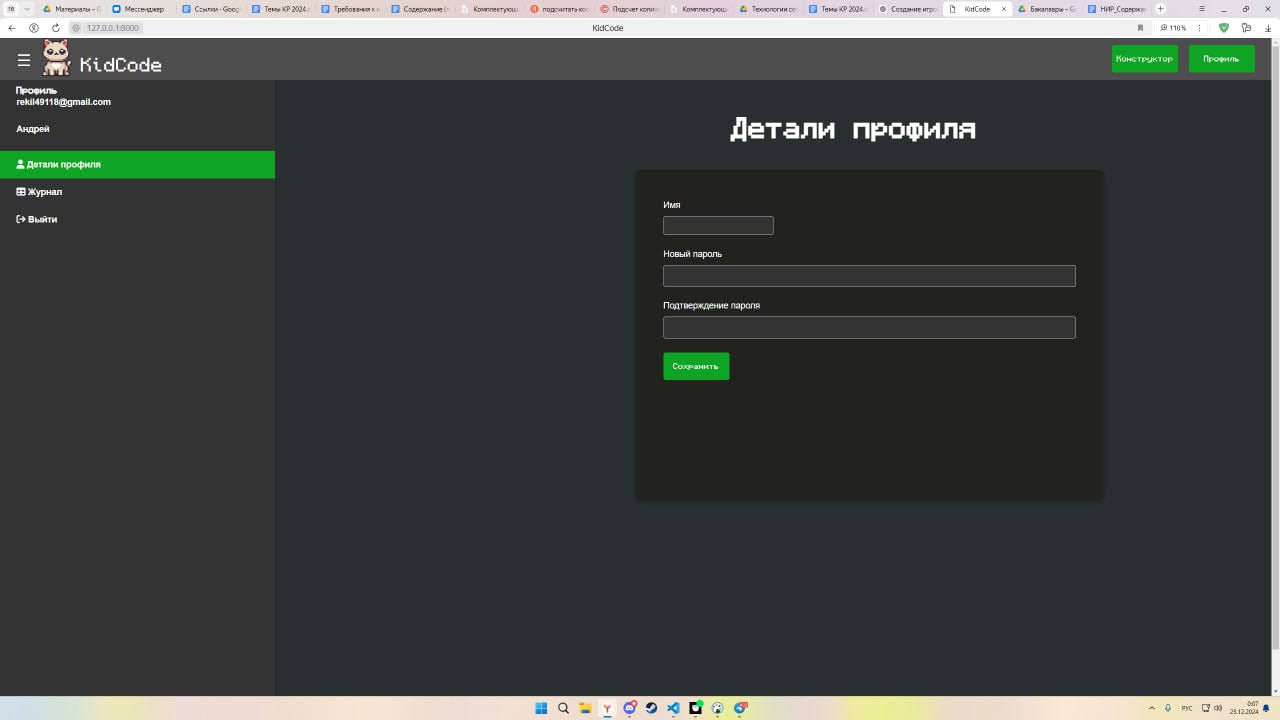
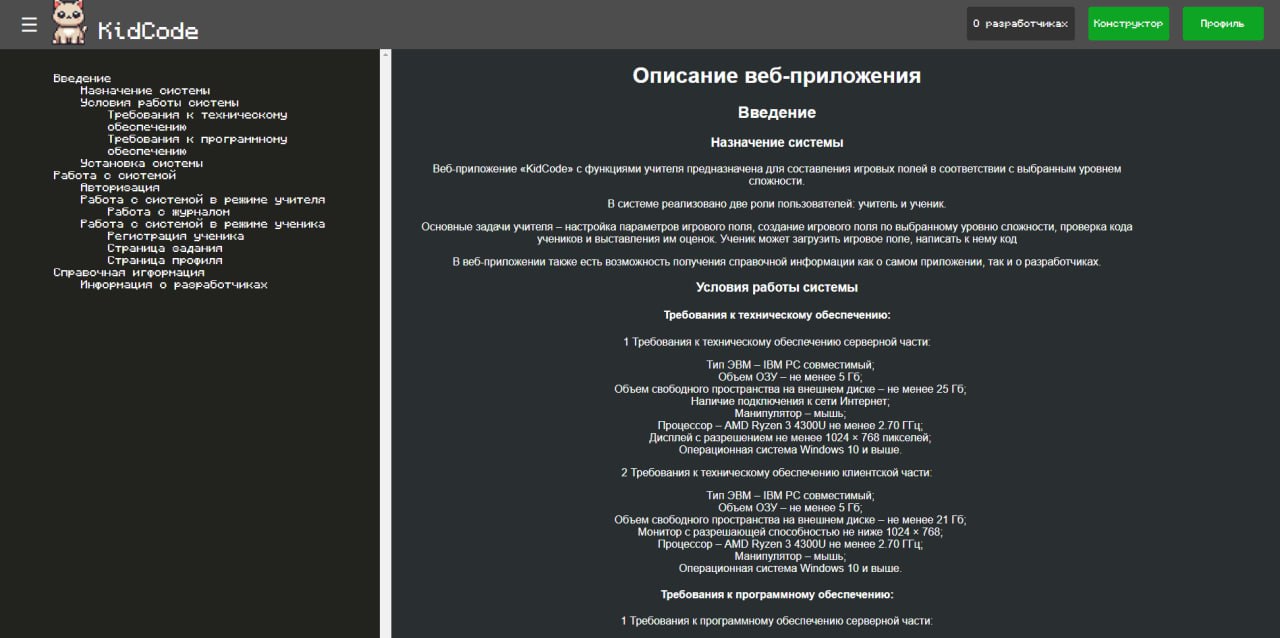


Рисунок А.5 – Страница профиля

Пользователь может посмотреть страницу «Справочная информация», нажав на соответствующую кнопку. На этой странице рассказывается о функциональных возможностях сайта, доступных пользователям, и описывает все ключевые функции и удобства, предлагаемые для обучения через веб-сервис. Вид страницы данного раздела приведён на рисунке А.6.

На рисунке А.7 приведено модальное окно «О разработчиках». Здесь пользователь может посмотреть сведения о команде, создавшей данное приложение.

  
Рисунок А.6 – Страница раздела «Справочная информация»

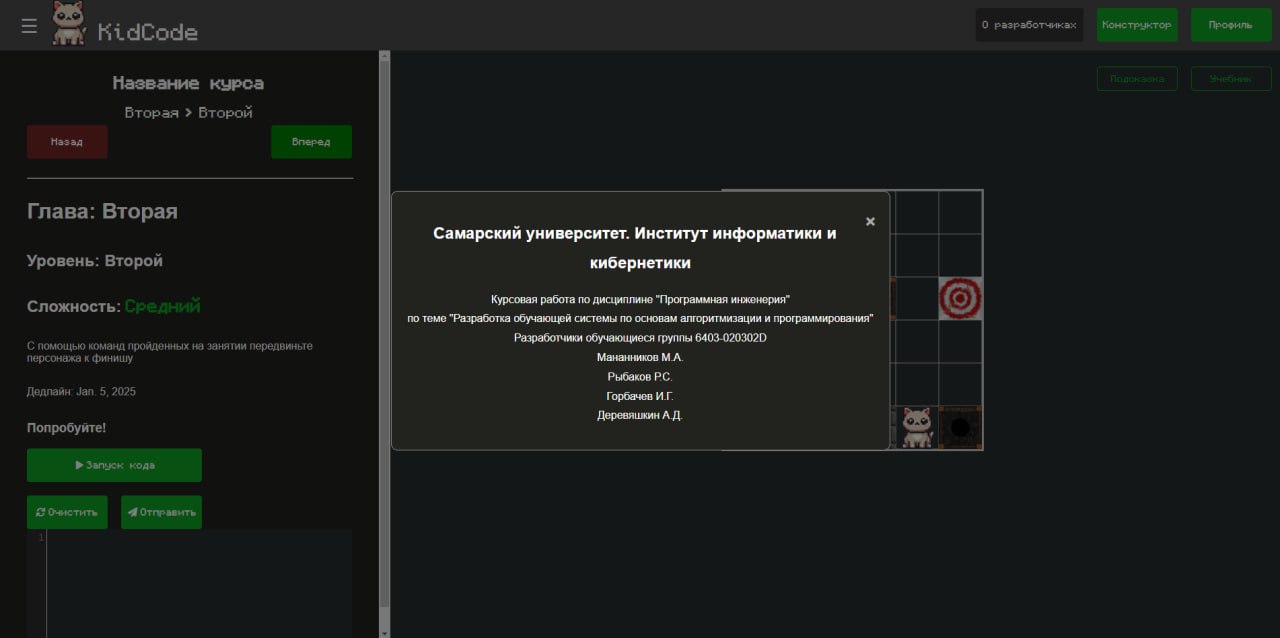


Рисунок А.7 – Модальное окно с информацией о разработчиках

А.4.1 Работа с системой в режиме учителя

Если был выполнен вход в роли учителя, переход на страницу личного профиля и открыт раздел «Журнал», который представлен на рисунке А.8, отобразится страница успеваемости учеников. Здесь учитель может применить фильтры для отображения интересующей его информации по полям «Статус», «Уровень» и «Имя».

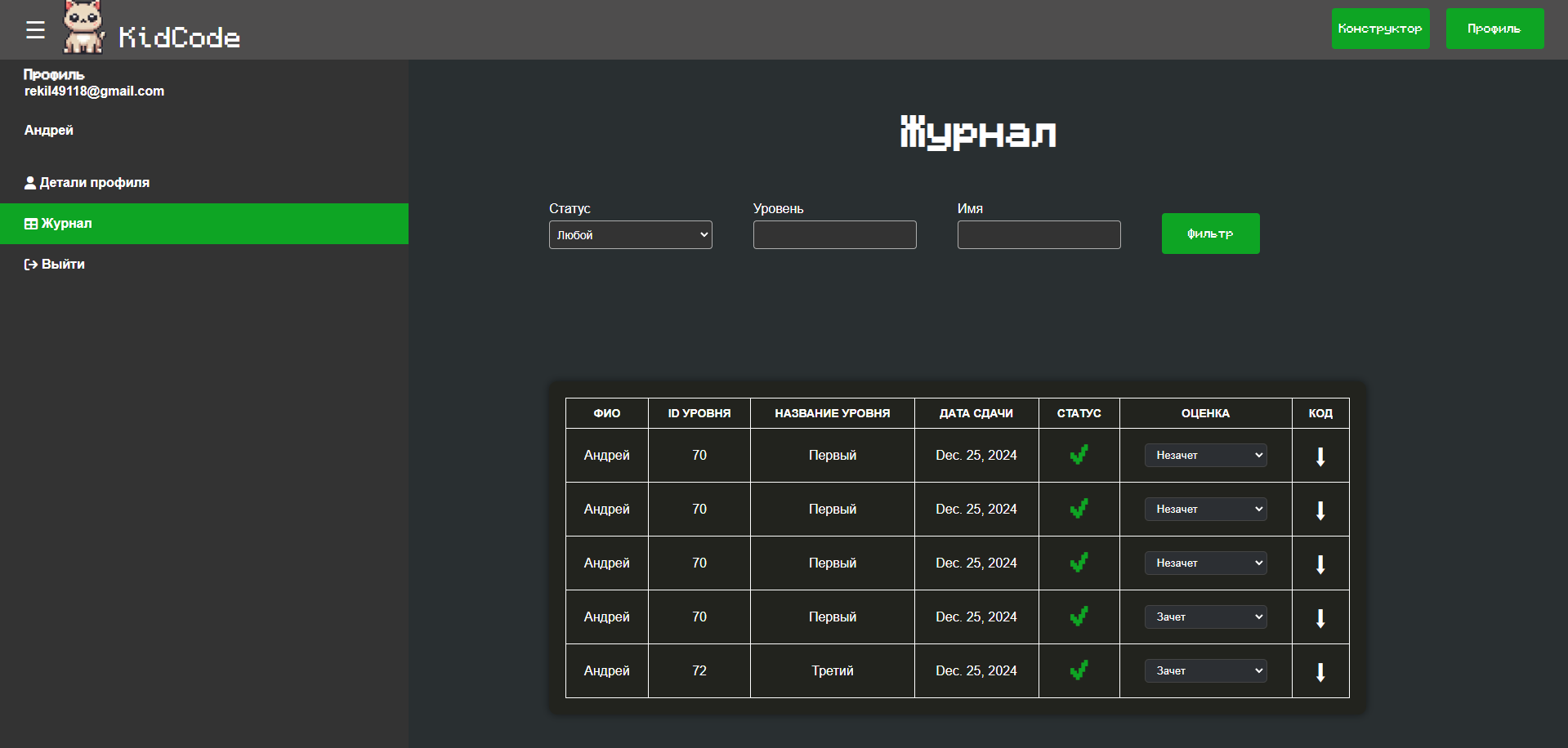


Рисунок А.8 – Страница журнала

На рисунке А.9 приведена страница конструктора. Здесь учитель создает задания для учеников. Учителю будет предложено два варианта создания задания: автоматически или вручную. При нажатии на кнопку «Автоматически» для генерации игрового поля учитель должен задать уровень сложности задания. Далее ввести текст задания и нажать кнопку «Сгенерировать». Если генерация прошла успешно, требуется нажать на кнопку «Сохранить». В случае, если поля введены некорректно, то учитель должен нажать кнопку «Очистить» для удаления данных в полях.

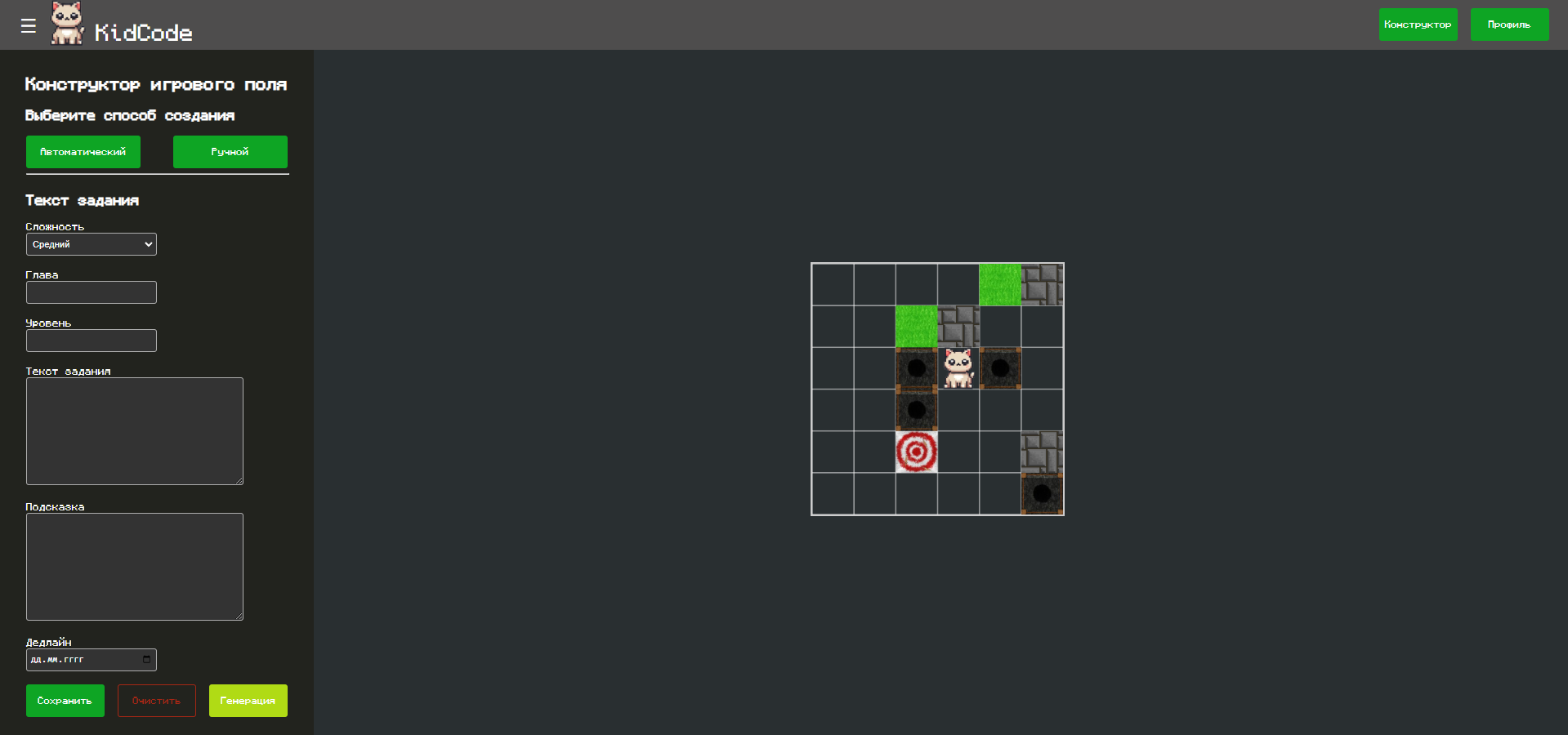


Рисунок А.9 – Страница конструктора в автоматическом режиме

На рисунке А.10 представлена страница создания задания в ручном режиме. По нажатию на кнопку «Вручную», учителю требуется ввести параметры игрового поля, выбрать шаблоны, выбрать уровень сложности и ввести текст задания. После заполнения полей, учителю требуется нажать кнопку «Сохранить». Если задание создано успешно, то задание сохраняется в БД. В случае, если поля введены некорректно, то учитель должен нажать кнопку «Очистить» для удаления данных в полях.

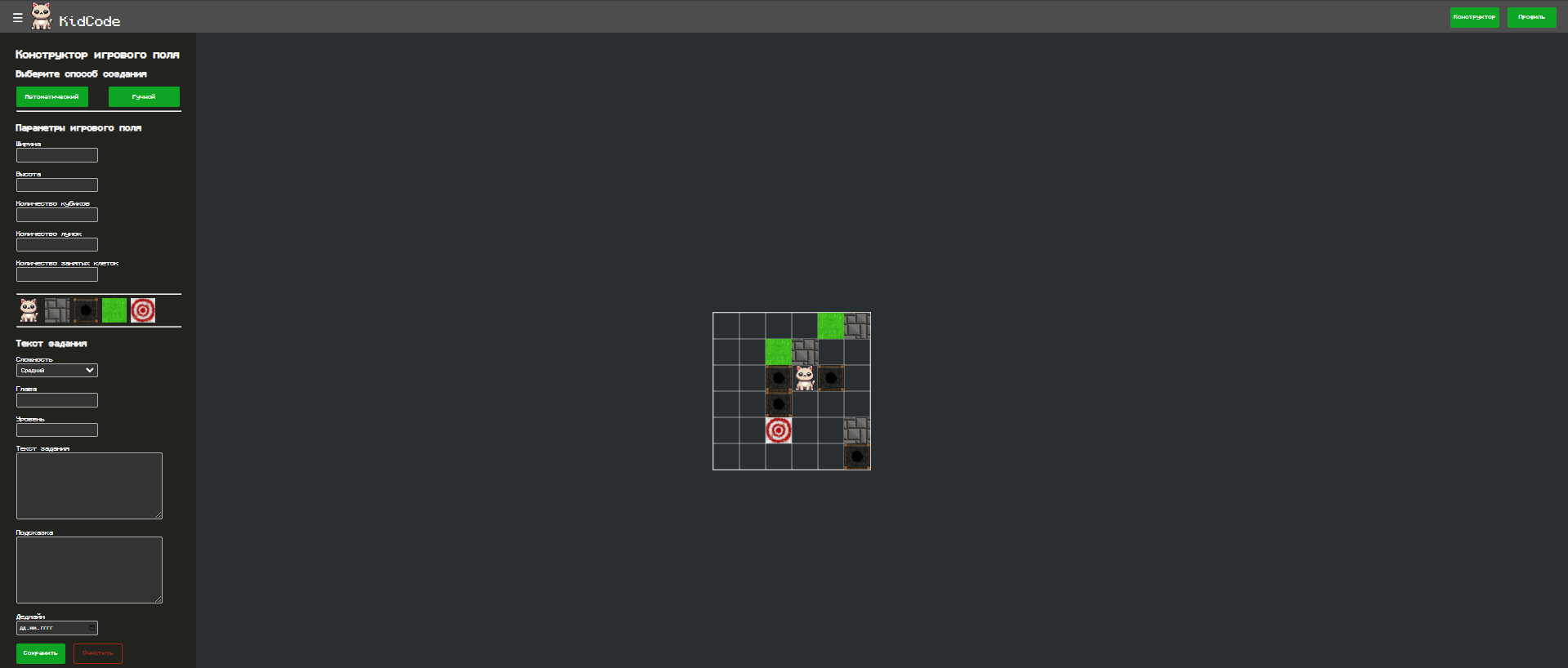


Рисунок А.10 – Страница конструктора в ручном режиме

ПРИЛОЖЕНИЕ Б   
Листинг модулей программы

document.addEventListener('DOMContentLoaded', () => {

let playerStart = { x: 0, y: 0 };

const canvas = document.getElementById('gameCanvas');

const ctx = canvas.getContext('2d');

const cellSize = 64;

let gridWidth = document.getElementById('id\_width');

let gridHeight = document.getElementById('id\_height');

canvas.width = gridWidth \* cellSize;

canvas.height = gridHeight \* cellSize;

//Drag and drop

const templates = document.querySelectorAll('.template');

let isDragging = false; // Флаг для отслеживания процесса перетаскивания

let currentTemplate = null; // Текущий шаблон, который перетаскивается

let offsetX = 0; // Смещение для корректного размещения

let offsetY = 0;

let placedObjects = []; // Массив для хранения размещённых объектов

const manualform = document.getElementById('manualForm');

const autoform = document.getElementById('autoForm');

const textform = document.getElementById('textForm');

const generateButton = document.getElementById('generateButton');

const difficultySelect = document.getElementById('id\_difficult');

const autoButton = document.getElementById('autoButton');

const savemanual = document.getElementById('savemanual');

const saveauto = document.getElementById('saveauto');

let randomData = {};

const cubeInput = document.getElementById('id\_cube'); // Поле "Количество кубиков"

const holeInput = document.getElementById('id\_hole'); // Поле "Количество лунок"

const blockInput = document.getElementById('id\_block'); // Поле "Количество занятых клеток"

// Переменные для отслеживания лимитов

let maxCubes = parseInt(cubeInput.value, 10) || 0;

let maxHoles = parseInt(holeInput.value, 10) || 0;

let maxBlocks = parseInt(blockInput.value, 10) || 0;

const manualButton = document.getElementById('manualButton');

const manualForm = document.getElementById('manualForm');

autoButton.addEventListener('click', () => {

clearField(); // Очищаем поле

manualForm.style.display = 'none'; // Скрываем ручную форму

generateButton.style.display = 'block';

saveauto.style.display = 'block';

savemanual.style.display = 'none';

});

manualButton.addEventListener('click', () => {

clearField(); // Очищаем поле

manualForm.style.display = 'block'; // Показываем ручную форму

generateButton.style.display = 'none';

saveauto.style.display = 'none';

savemanual.style.display = 'block';

});

// Слушатели для обновления лимитов при изменении полей ввода

cubeInput.addEventListener('input', () => {

maxCubes = parseInt(cubeInput.value, 10) || 0;

});

holeInput.addEventListener('input', () => {

maxHoles = parseInt(holeInput.value, 10) || 0;

});

blockInput.addEventListener('input', () => {

maxBlocks = parseInt(blockInput.value, 10) || 0;

});

function clearField() {

placedObjects = []; // Очищаем массив объектов

gridWidth.value = 4; // Минимальное значение ширины

gridHeight.value = 4; // Минимальное значение высоты

canvas.width = 4 \* cellSize; // Устанавливаем минимальный размер canvas

canvas.height = 4 \* cellSize;

drawGrid(); // Перерисовываем сетку

}

generateButton.addEventListener("click", function () {

const selectedDifficulty = difficultySelect.value; // Получаем выбранный уровень

if (selectedDifficulty === 'easy') {

randomData = {

'width': 4,

'height': 4,

'cube': 2,

'hole': 3,

'block': 1,

};

} else if (selectedDifficulty === 'medium') {

randomData = {

'width': 6,

'height': 6,

'cube': 3,

'hole': 4,

'block': 2,

};

} else if (selectedDifficulty === 'hard') {

randomData = {

'width': 8,

'height': 8,

'cube': 5,

'hole': 6,

'block': 3,

};

}

gridWidth = randomData['width'];

gridHeight = randomData['height'];

canvas.width = gridWidth \* cellSize;

canvas.height = gridHeight \* cellSize;

placedObjects = generateObjects(randomData); // Генерируем объекты

drawGrid(); // Перерисовываем сетку

});

function generateObjects(randomData) {

const generatedObjects = []; // Массив для хранения объектов

const addObject = (type, count) => {

let placed = 0;

while (placed < count) {

const x = Math.floor(Math.random() \* gridWidth) \* cellSize;

const y = Math.floor(Math.random() \* gridHeight) \* cellSize;

// Проверяем, занята ли клетка

const isOccupied = generatedObjects.some(obj => obj.x === x && obj.y === y);

if (!isOccupied) {

generatedObjects.push({

image: (() => {

const img = new Image();

if (type === 'cube') img.src = cubeImage;

else if (type === 'hole') img.src = holeImage;

else if (type === 'block') img.src = blockImage;

else if (type == 'player') img.src = playerImage;

else if (type == 'goal') img.src = goalImage;

return img;

})(),

x: x,

y: y,

id: type

});

placed++;

}

}

};

// Добавляем объекты по типу

addObject('goal', 1)

addObject('cube', randomData.cube); // Кубики

addObject('hole', randomData.hole); // Лунки

addObject('block', randomData.block); // Блоки

addObject('player', 1);

return generatedObjects;

}

function drawGrid() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.strokeStyle = "#FFFFFF";

for (let x = 0; x <= canvas.width; x += cellSize) {

ctx.beginPath();

ctx.moveTo(x, 0);

ctx.lineTo(x, canvas.height);

ctx.stroke();

}

for (let y = 0; y <= canvas.height; y += cellSize) {

ctx.beginPath();

ctx.moveTo(0, y);

ctx.lineTo(canvas.width, y);

ctx.stroke();

}

placedObjects.forEach(obj => {

ctx.drawImage(obj.image, obj.x, obj.y, cellSize, cellSize);

});

}

function updateCanvasSize() {

const newGridWidth = parseInt(document.getElementById('id\_width').value, 10); // Считываем значение ширины

const newGridHeight = parseInt(document.getElementById('id\_height').value, 10); // Считываем значение высоты

if (isNaN(newGridWidth) || isNaN(newGridHeight)) {

return;

}

if (newGridHeight > 10 || newGridWidth > 10 || newGridHeight < 4 || newGridWidth < 4) {

alert('Допустимый диапазон значений от 4 до 10. Попробуйте снова');

return;

}

canvas.width = newGridWidth \* cellSize;

canvas.height = newGridHeight \* cellSize;

drawGrid();

}

function getObjectAtPosition(x, y) {

return placedObjects.find(obj =>

x >= obj.x &&

x < obj.x + cellSize &&

y >= obj.y &&

y < obj.y + cellSize

);

}

// Перетаскивание шаблона из списка

templates.forEach(template => {

template.addEventListener('mousedown', e => {

const rect = template.getBoundingClientRect();

currentTemplate = {

image: new Image(),

x: 0,

y: 0,

id: template.dataset.type // Получаем тип объекта из data-type

};

currentTemplate.image.src = template.src;

offsetX = e.clientX - rect.left;

offsetY = e.clientY - rect.top;

isDragging = true;

});

});

// Начало перетаскивания на canvas

canvas.addEventListener('mousedown', e => {

const rect = canvas.getBoundingClientRect();

const mouseX = e.clientX - rect.left;

const mouseY = e.clientY - rect.top;

const selectedObject = getObjectAtPosition(mouseX, mouseY);

if (selectedObject) {

currentTemplate = selectedObject;

isDragging = true;

offsetX = mouseX - selectedObject.x;

offsetY = mouseY - selectedObject.y;

// Удаляем объект из массива, чтобы избежать дублирования

const index = placedObjects.indexOf(selectedObject);

if (index > -1) placedObjects.splice(index, 1);

}

});

// Перемещение объекта

canvas.addEventListener('mousemove', e => {

if (isDragging && currentTemplate) {

const rect = canvas.getBoundingClientRect();

const mouseX = e.clientX - rect.left;

const mouseY = e.clientY - rect.top;

currentTemplate.x = Math.floor((mouseX - offsetX) / cellSize) \* cellSize;

currentTemplate.y = Math.floor((mouseY - offsetY) / cellSize) \* cellSize;

drawGrid();

ctx.drawImage(

currentTemplate.image,

currentTemplate.x,

currentTemplate.y,

cellSize,

cellSize

);

}

});

// Завершение перетаскивания

canvas.addEventListener('mouseup', () => {

if (isDragging && currentTemplate) {

// Подсчитываем количество объектов текущего типа

const objectCount = placedObjects.filter(obj => obj.id === currentTemplate.id).length;

let limitExceeded = false;

if (currentTemplate.id === 'cube' && objectCount >= maxCubes) {

alert('Превышено количество кубиков!');

limitExceeded = true;

} else if (currentTemplate.id === 'hole' && objectCount >= maxHoles) {

alert('Превышено количество лунок!');

limitExceeded = true;

} else if (currentTemplate.id === 'block' && objectCount >= maxBlocks) {

alert('Превышено количество блоков!');

limitExceeded = true;

}

else if (currentTemplate.id === 'player' && objectCount >= 1) {

alert('Превышено количество игроков!');

limitExceeded = true;

}

else if (currentTemplate.id === 'goal' && objectCount >= 1) {

alert('Превышено количество целей!');

limitExceeded = true;

}

if (!limitExceeded) {

// Проверяем, занята ли клетка

const isOccupied = placedObjects.some(obj =>

obj.x === currentTemplate.x && obj.y === currentTemplate.y

);

if (!isOccupied) {

// Добавляем объект в массив, если клетка не занята и лимит не превышен

placedObjects.push({

image: currentTemplate.image,

x: currentTemplate.x,

y: currentTemplate.y,

id: currentTemplate.id

});

} else {

alert('Эта клетка уже занята!');

}

}

currentTemplate = null;

isDragging = false;

drawGrid(); // Перерисовываем сетку

}

});

// Событие для удаления объекта

canvas.addEventListener('dblclick', (e) => {

const rect = canvas.getBoundingClientRect();

const mouseX = e.clientX - rect.left;

const mouseY = e.clientY - rect.top;

const objectToRemove = getObjectAtPosition(mouseX, mouseY);

if (objectToRemove) {

// Удаляем объект из массива

const index = placedObjects.indexOf(objectToRemove);

if (index > -1) {

placedObjects.splice(index, 1);

}

drawGrid(); // Перерисовываем сетку

}

});

gridWidth.addEventListener('blur', updateCanvasSize);

gridHeight.addEventListener('blur', updateCanvasSize);

drawGrid();

//Функция для обработки ручного поля

savemanual.addEventListener('click', function (e) {

e.preventDefault(); // Отключаем стандартное поведение формы

if (placedObjects.length === 0) {

alert('Поле пустое! Пожалуйста, добавьте объекты перед сохранением.');

return; // Останавливаем выполнение функции

}

// Собираем данные из формы

const formData = new FormData(textform);

const formData2 = new FormData(manualform);

const jsonData = {};

formData.forEach((value, key) => {

jsonData[key] = value;

});

formData2.forEach((value, key) => {

jsonData[key] = value;

});

// Добавляем дополнительные данные (например, координаты из canvas)

jsonData.data = placedObjects;

console.log(jsonData);

// Отправляем данные через fetch

fetch('/constructor/', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRFToken': formData.get('csrfmiddlewaretoken') // Передаём CSRF токен

},

body: JSON.stringify(jsonData)

})

.then(response => {

if (!response.ok) { // Проверяем, успешен ли ответ

return response.json().then(errData => {

throw new Error(errData.error || 'Ошибка при сохранении'); // Генерируем ошибку с сообщением

});

}

return response.json(); // Возвращаем JSON-ответ

})

.then(data => {

console.log(data);

alert('Сохранение прошло успешно!');

})

.catch(error => {

console.error('Ошибка:', error);

alert(error.message);

});

});

//Функция для обработки автоматического поля

saveauto.addEventListener('click', function (e) {

e.preventDefault(); // Отключаем стандартное поведение формы

if (placedObjects.length === 0) {

alert('Поле пустое! Пожалуйста, добавьте объекты перед сохранением.');

return; // Останавливаем выполнение функции

}

// Собираем данные из формы

const formData = new FormData(textform);

const jsonData = {};

formData.forEach((value, key) => {

jsonData[key] = value;

});

Object.assign(jsonData, randomData);

jsonData.data = placedObjects;

jsonData.playerStart = playerStart;

// Отправляем данные через fetch

fetch('/constructor/', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRFToken': formData.get('csrfmiddlewaretoken') // Передаём CSRF токен

},

body: JSON.stringify(jsonData)

})

.then(response => {

if (!response.ok) { // Проверяем, успешен ли ответ

return response.json().then(errData => {

throw new Error(errData.error || 'Ошибка при сохранении'); // Генерируем ошибку с сообщением

});

}

return response.json(); // Возвращаем JSON-ответ

})

.then(data => {

console.log(data);

alert('Сохранение прошло успешно!');

})

.catch(error => {

console.error('Ошибка:', error);

alert('Произошла ошибка при сохранении: ' + error.message); // Выводим сообщение об ошибке

});

});

});

from datetime import datetime

from django.db import connection

from django.urls import reverse\_lazy

from django.views.decorators.http import require\_POST

from django.contrib.auth.decorators import login\_required

from django.shortcuts import redirect, render, get\_object\_or\_404

from django.utils import timezone

from django.db.models import Q

from django.contrib import messages

from django.http import JsonResponse

from django.contrib.auth.views import PasswordChangeView

from django.views.generic import View,UpdateView, TemplateView

from django.views.decorators.csrf import csrf\_exempt

import json

from .forms import \*

from mainapp.models import \*

from mainapp.mixins import RoleRequiredMixin

from users.models import User

from RestrictedPython import compile\_restricted, safe\_globals, utility\_builtins

def about(request):

return render(request, 'mainapp/about.html')

def get\_sended\_task(request, pk, user\_id):

# Загрузка задачи

task = get\_object\_or\_404(Task, pk=pk)

print("dssdds")

# Используем связь task.gamefield\_id для получения игрового поля

game\_field = get\_object\_or\_404(GameField, id=task.gamefield\_id)

next\_task = Task.objects.filter(id\_\_gt=task.id).order\_by('id').first() # Задача с большим id

prev\_task = Task.objects.filter(id\_\_lt=task.id).order\_by('-id').first() # Задача с меньшим id

# Инициализация начальных координат игрока по умолчанию

initial\_x, initial\_y = 0, 0

# Попытка найти игрока в данных игрового поля

try:

game\_field\_data = game\_field.data # Предполагается, что это уже список

if isinstance(game\_field\_data, str):

game\_field\_data = json.loads(game\_field\_data) # Преобразование из строки JSON, если это строка

player\_data = next((item for item in game\_field\_data if item.get('id') == 'player'), None)

if player\_data:

initial\_x = player\_data.get('x', 0)

initial\_y = player\_data.get('y', 0)

except Exception as e:

print(f"Ошибка при обработке данных игрового поля: {e}")

# Извлекаем код из базы данных, если он существует

try:

code\_entry = Code.objects.get(user=user\_id, game\_field=game\_field)

user\_code = code\_entry.code # Получаем код

except Code.DoesNotExist:

user\_code = "" # Если кода нет, пустое значение

# Проверяем, существует ли Player для данного пользователя и игрового поля

player, created = Player.objects.get\_or\_create(

user=user\_id,

game\_field=game\_field,

defaults={'x': initial\_x, 'y': initial\_y} # Используем извлеченные координаты

)

if created:

print(f"Создан новый игрок для пользователя {request.user.id} и игрового поля {game\_field.id} с координатами ({initial\_x}, {initial\_y})")

# Передаем данные в контекст

context = {

'task': task,

'game\_field': game\_field,

'player': player,

'user\_code': user\_code, # Добавляем код в контекст

'next\_task': next\_task, # Следующая задача

'prev\_task': prev\_task, # Предыдущая задача

}

print(context)

return render(request, 'mainapp/task.html', context)

class Task1(View):

template\_name = 'mainapp/task.html'

def get(self, request, pk):

if request.path.endswith('/data/'):

return self.get\_game\_field\_data(request, pk)

elif request.path.endswith('/clue/'):

return self.get\_clue(request, pk)

elif request.path.endswith('/position/'):

return self.get\_player\_position(request, pk, request.user.id)

else:

return self.get\_task\_view(request, pk)

def post(self, request, pk, user\_id):

if 'reset' in request.path:

return self.post\_reset(request, pk, user\_id)

elif 'submit\_grade' in request.path: # Обработка отправки оценки

return self.submit\_grade(request, pk, user\_id)

else:

return self.post\_move\_player(request, pk, user\_id)

def get\_task\_view(self, request, pk):

# Загрузка задачи

task = get\_object\_or\_404(Task, pk=pk)

print(task)

# Используем связь task.gamefield\_id для получения игрового поля

game\_field = get\_object\_or\_404(GameField, id=task.gamefield\_id)

next\_task = Task.objects.filter(id\_\_gt=task.id).order\_by('id').first() # Задача с большим id

prev\_task = Task.objects.filter(id\_\_lt=task.id).order\_by('-id').first() # Задача с меньшим id

# Инициализация начальных координат игрока по умолчанию

initial\_x, initial\_y = 0, 0

# Попытка найти игрока в данных игрового поля

try:

game\_field\_data = game\_field.data # Предполагается, что это уже список

if isinstance(game\_field\_data, str):

game\_field\_data = json.loads(game\_field\_data) # Преобразование из строки JSON, если это строка

player\_data = next((item for item in game\_field\_data if item.get('id') == 'player'), None)

if player\_data:

initial\_x = player\_data.get('x', 0)

initial\_y = player\_data.get('y', 0)

except Exception as e:

print(f"Ошибка при обработке данных игрового поля: {e}")

# Проверяем, существует ли Player для данного пользователя и игрового поля

player, created = Player.objects.get\_or\_create(

user=request.user,

game\_field=game\_field,

defaults={'x': initial\_x, 'y': initial\_y} # Используем извлеченные координаты

)

if created:

print(f"Создан новый игрок для пользователя {request.user.id} и игрового поля {game\_field.id} с координатами ({initial\_x}, {initial\_y})")

# Извлекаем код из базы данных, если он существует

try:

code\_entry = Code.objects.get(user=request.user, game\_field=game\_field)

user\_code = code\_entry.code # Получаем код

except Code.DoesNotExist:

user\_code = "" # Если кода нет, пустое значение

# Передаем данные в контекст

context = {

'task': task,

'game\_field': game\_field,

'player': player,

'user\_code': user\_code, # Добавляем код в контекст

'next\_task': next\_task, # Следующая задача

'prev\_task': prev\_task, # Предыдущая задача

}

return render(request, self.template\_name, context)

def get\_game\_field\_data(self, request, pk):

"""

Обработка AJAX-запросов для получения данных игрового поля.

"""

# Загружаем данные задачи

task = get\_object\_or\_404(Task, pk=pk)

# Используем связь task.gamefield\_id для получения игрового поля

game\_field = get\_object\_or\_404(GameField, id=task.gamefield\_id)

# Формируем JSON-ответ

return JsonResponse({

'data': game\_field.data,

'width': game\_field.width,

'height': game\_field.height,

})

def get\_clue(self, request, pk):

"""

Возвращает подсказку задачи в формате JSON.

"""

task = get\_object\_or\_404(Task, pk=pk)

return JsonResponse({'clue': task.clue})

def get\_player\_position(request, pk, user\_id):

try:

player = Player.objects.get(user\_id=user\_id)

return JsonResponse({"x": player.x, "y": player.y})

except Player.DoesNotExist:

return JsonResponse({"error": "Player not found"}, status=404)

def post\_move\_player(self, request, pk, user\_id):

print(f"Task ID: {pk}, Player ID: {user\_id}")

print(f"Request Body: {request.body.decode('utf-8')}")

# Получаем задачу и игровое поле

try:

task = get\_object\_or\_404(Task, pk=pk)

game\_field = get\_object\_or\_404(GameField, id=task.gamefield\_id)

except Exception as e:

print(f"Error fetching task or game field: {e}")

return JsonResponse({"error": "Task or game field not found"}, status=404)

# Пытаемся получить игрока

try:

player = Player.objects.get(user\_id=user\_id, game\_field=game\_field)

except Player.DoesNotExist:

return JsonResponse({"error": "Player not found"}, status=404)

# Получаем и проверяем пользовательский код

try:

data = json.loads(request.body)

user\_code = data.get('code', '')

print(f"Received code: {user\_code}")

except json.JSONDecodeError as e:

print(f"Error parsing JSON: {e}")

return JsonResponse({"error": "Некорректный формат данных JSON"}, status=400)

if not user\_code:

print("Code is empty")

return JsonResponse({"error": "Код не предоставлен"}, status=400)

# Сохраняем код в базу данных

try:

code\_entry, created = Code.objects.update\_or\_create(

user\_id=user\_id, game\_field=game\_field,

defaults={'code': user\_code}

)

if created:

print(f"Создана новая запись кода для пользователя {user\_id} на игровом поле {game\_field.id}")

else:

print(f"Обновлена запись кода для пользователя {user\_id} на игровом поле {game\_field.id}")

except Exception as e:

print(f"Error saving code: {e}")

return JsonResponse({"error": "Ошибка при сохранении кода"}, status=500)

# Настраиваем безопасные переменные для исполнения кода

safe\_globals = {

"\_getiter\_": iter,

\*\*utility\_builtins,

}

safe\_locals = {

"move\_down": lambda: self.move\_player(player, 0, 64, game\_field),

"move\_up": lambda: self.move\_player(player, 0, -64, game\_field),

"move\_left": lambda: self.move\_player(player, -64, 0, game\_field),

"move\_right": lambda: self.move\_player(player, 64, 0, game\_field),

"range": range,

}

# Исполнение пользовательского кода

try:

byte\_code = compile\_restricted(user\_code, '<inline>', 'exec')

print(f"Executing user code: {user\_code}")

print(f"Safe locals before execution: {safe\_locals.keys()}")

exec(byte\_code, {\*\*safe\_globals, \*\*utility\_builtins}, safe\_locals)

player.refresh\_from\_db() # Убедимся, что данные обновились

except ValueError as e:

# Ловим ошибки перемещения и передаем их клиенту

print(f"Ошибка выполнения кода: {e}")

return JsonResponse({"error": str(e)}, status=400)

except Exception as e:

print(f"Ошибка выполнения кода: {e}")

return JsonResponse({"error": str(e)}, status=400)

# Проверка достижения цели (goal)

level\_completed = False

for obj in game\_field.data:

if obj["x"] == player.x and obj["y"] == player.y and obj["id"] == "goal":

print("Player reached the goal!")

level\_completed = True

break

response\_data = {

"x": player.x,

"y": player.y,

"level\_completed": level\_completed,

}

if level\_completed:

response\_data["message"] = "Уровень пройден!"

print(f"Player coordinates: x={player.x}, y={player.y}, level\_completed={level\_completed}")

return JsonResponse(response\_data)

def move\_player(self, player, dx, dy, game\_field):

new\_x = player.x + dx

new\_y = player.y + dy

print(f"Attempting to move player: current=({player.x}, {player.y}), new=({new\_x}, {new\_y})")

print(f"Game field dimensions: width={game\_field.width}, height={game\_field.height}")

if new\_x < 0:

raise ValueError("Нельзя выйти за левую границу игрового поля")

if new\_x >= game\_field.width \* 64:

raise ValueError("Нельзя выйти за правую границу игрового поля")

if new\_y < 0:

raise ValueError("Нельзя выйти за верхнюю границу игрового поля")

if new\_y >= game\_field.height \* 64:

raise ValueError("Нельзя выйти за нижнюю границу игрового поля")

if self.is\_cell\_occupied(new\_x, new\_y, game\_field):

raise ValueError("Клетка занята")

player.x = new\_x

player.y = new\_y

player.save()

print(f"Player moved to: ({player.x}, {player.y})")

def is\_cell\_occupied(self, x, y, game\_field):

"""

Проверяет, занята ли клетка с координатами (x, y), за исключением объектов с id "goal".

:param x: Координата X клетки

:param y: Координата Y клетки

:param game\_field: Объект игрового поля, содержащий список объектов в поле `data`

:return: True, если клетка занята объектом, на который нельзя перемещаться, иначе False

"""

for obj in game\_field.data:

if obj["x"] == x and obj["y"] == y:

if obj["id"] == "goal" or obj["id"] == "player":

# Разрешаем перемещение на объект с id "goal"

print(f"Cell contains goal: ({x}, {y})")

return False

else:

# Блокируем перемещение на все остальные объекты

print(f"Cell occupied by object: {obj['id']} at ({x}, {y})")

return True

return False

def post\_reset(self, request, pk, user\_id):

print(f"Resetting data for Task ID: {pk}, Player ID: {user\_id}")

# Получаем задачу и игровое поле

try:

task = get\_object\_or\_404(Task, pk=pk)

game\_field = get\_object\_or\_404(GameField, id=task.gamefield\_id)

except Exception as e:

print(f"Error fetching task or game field: {e}")

return JsonResponse({"error": "Task or game field not found"}, status=404)

# Пытаемся получить игрока

try:

player = Player.objects.get(user\_id=user\_id, game\_field=game\_field)

except Player.DoesNotExist:

return JsonResponse({"error": "Player not found"}, status=404)

# Извлекаем данные игрового поля

try:

game\_field\_data = game\_field.data # Предполагается, что это уже список

if isinstance(game\_field\_data, str):

game\_field\_data = json.loads(game\_field\_data) # Преобразуем строку в список

# Находим объект игрока в данных игрового поля

player\_data = next((item for item in game\_field\_data if item.get('id') == 'player'), None)

if player\_data:

# Если нашли игрока, сбрасываем его координаты

initial\_x = player\_data.get('x', 0) # Берем координаты из данных

initial\_y = player\_data.get('y', 0)

else:

initial\_x, initial\_y = 0, 0 # Если игрока в данных нет, используем 0, 0

except Exception as e:

print(f"Ошибка при обработке данных игрового поля: {e}")

initial\_x, initial\_y = 0, 0 # Значения по умолчанию, если не удалось обработать данные

# Сбрасываем координаты игрока

player.x = initial\_x

player.y = initial\_y

player.save()

print(f"Player {user\_id} coordinates reset to ({initial\_x}, {initial\_y})")

# Сбрасываем код

try:

code\_entry = Code.objects.get(user\_id=user\_id, game\_field=game\_field)

code\_entry.code = "" # Очищаем код

code\_entry.save()

print(f"Code for player {user\_id} cleared.")

except Code.DoesNotExist:

print(f"No code found for player {user\_id}. Nothing to clear.")

return JsonResponse({'status': 'success', 'message': 'Player and code reset successfully'})

def submit\_grade(self, request, pk, user\_id):

# Проверяем, что это POST-запрос с JSON данными

if request.method != 'POST':

return JsonResponse({'error': 'Неверный метод запроса'}, status=405)

try:

data = json.loads(request.body)

except json.JSONDecodeError:

return JsonResponse({'error': 'Некорректный формат данных'}, status=400)

# Получаем оценку и статус из данных POST-запроса

grade = data.get('grade', 'fail') # Значение по умолчанию 'fail'

status = data.get('status', 'sended') # Значение по умолчанию 'sended'

if not grade or grade not in ['pass', 'fail']:

return JsonResponse({'error': 'Неверно указана оценка'}, status=400)

if not status or status not in ['sended', 'not\_sended']:

return JsonResponse({'error': 'Неверно указан статус'}, status=400)

# Получаем задачу

try:

task = get\_object\_or\_404(Task, pk=pk)

print(task)

except Task.DoesNotExist:

return JsonResponse({'error': 'Задача не найдена'}, status=404)

# Получаем запись кода для пользователя и задачи

code\_entry = Code.objects.get(user\_id=user\_id, game\_field=task.gamefield)

if not code\_entry.code.strip(): # Используем strip() для удаления пробелов

return JsonResponse({'error': 'Код не найден или пуст для этой задачи и пользователя'}, status=404)

# Создаем запись в модели Grade

grade\_entry = Grade(

task=task,

user\_id=user\_id,

code=code\_entry,

grade=grade,

status=status,

submission\_date=timezone.now()

)

grade\_entry.save()

return JsonResponse({'status': 'success', 'message': 'Оценка успешно отправлена'})

@login\_required

@require\_POST

def update\_grade(request):

if 'grade\_id' not in request.POST or 'new\_grade' not in request.POST:

return JsonResponse({'error': 'grade\_id and new\_grade are required'}, status=400)

grade\_id = request.POST.get('grade\_id')

new\_grade = request.POST.get('new\_grade')

with connection.cursor() as cursor:

try:

cursor.execute("UPDATE mainapp\_grade SET grade = %s WHERE id = %s", [new\_grade, grade\_id])

return JsonResponse({'status': 'success'})

except Exception as e:

return JsonResponse({'error': str(e)}, status=500)

def record(request):

record\_data = []

form = RecordForm(request.GET)

user\_id = request.user.id

filters = Q(user\_id=user\_id)

if form.is\_valid():

grade = form.cleaned\_data.get('grade')

level = form.cleaned\_data.get('level')

if grade:

filters &= Q(grade=grade)

if level:

filters &= Q(level=level)

record\_data = list(RecordView.objects.filter(filters))

else:

record\_data = list(RecordView.objects.filter(filters))

if record\_data == []:

messages.error(request, "По вашему запросу ничего не найдено")

return render(request, 'mainapp/record.html', {'record\_data': record\_data, 'form': form})

class Journal(RoleRequiredMixin, TemplateView):

form\_class = JournalForm

template\_name = 'mainapp/journal.html'

success\_url = reverse\_lazy("mainapp:journal")

def get(self, request):

journal\_data = []

form = JournalForm(request.GET)

if form.is\_valid():

status = form.cleaned\_data.get('status')

level = form.cleaned\_data.get('level')

user\_name = form.cleaned\_data.get('name')

filters = Q()

if status:

filters &= Q(status=status)

if level:

filters &= Q(level=level)

if user\_name:

filters &= Q(name\_\_iexact=user\_name)

journal\_data = list(JournalView.objects.filter(filters))

else:

journal\_data = list(JournalView.objects.all())

if journal\_data == []:

messages.error(request, "По вашему запросу ничего не найдено")

return render(request,'mainapp/journal.html', {'journal\_data': journal\_data, 'form': form})

def index(request):

return redirect('users/login')

class UserProfileChangeView(View):

form\_class = UserProfileChangeForm

template\_name = "mainapp/profile.html"

def get(self, request, \*args, \*\*kwargs):

form = self.form\_class()

return render(request, self.template\_name, {'form': form})

def post(self, request, \*args, \*\*kwargs):

form = self.form\_class(request.POST)

if form.is\_valid():

# Смена имени

name = form.cleaned\_data.get('name')

if name:

request.user.name = name

request.user.save()

messages.success(request, "Имя успешно изменено!")

else:

messages.error(request, "Заполните поле имени!")

# Смена пароля

new\_password1 = form.cleaned\_data.get('new\_password1')

new\_password2 = form.cleaned\_data.get('new\_password2')

if new\_password1: # Проверяем только если новое имя пароля заполнено

if new\_password1 == new\_password2:

request.user.set\_password(new\_password1)

request.user.save()

messages.success(request, "Пароль успешно изменён!")

else:

messages.error(request, "Пароли не совпадают!")

return redirect('mainapp:profile')

return render(request, self.template\_name, {'form': form})

class FieldsSettings(View):

template\_name = 'mainapp/constructor.html'

model = Task

fields = []

def get(self, request, \*args, \*\*kwargs):

fields\_form = FieldsSettingsForm()

task\_form = TaskTextForm()

return render(request, self.template\_name, {

'fields\_form': fields\_form,

'task\_form': task\_form

})

def post(self, request, \*args, \*\*kwargs):

data = json.loads(request.body) # Загружаем данные из JSON

has\_player = False

has\_goal = False

for obj in data.get('data', []): # Перебираем массив data

obj\_id = obj.get('id')

if obj\_id == 'player':

has\_player = True

elif obj\_id == 'goal':

has\_goal = True

if not has\_player:

return JsonResponse({'error': 'На поле должен быть персонаж'}, status=400)

if not has\_goal:

return JsonResponse({'error': 'На поле должен быть финиш'}, status=400)

fields\_form = FieldSaveForm(data)

task\_form = TaskTextForm(data)

try:

# Преобразуем строку дедлайна в объект datetime

deadline = datetime.strptime(data['deadline'], '%Y-%m-%d')

except ValueError:

return JsonResponse({'error': 'Некорректный формат даты дедлайна'}, status=400)

# Проверяем, что дедлайн больше текущей даты

if deadline <= datetime.now():

print(data['deadline'])

return JsonResponse({'error': 'Дедлайн должен быть больше текущей даты'}, status=400)

if fields\_form.is\_valid() and task\_form.is\_valid():

game\_field = fields\_form.save()

task = task\_form.save(commit=False)

task.gamefield = game\_field

task.save()

# Извлечение данных playerStart и сохранение в модель Player

player\_data = None

for obj in data.get('data', []): # Перебираем массив `data`

if obj.get('id') == 'player': # Ищем объект с id: player

player\_data = obj

break

if player\_data:

# Сохраняем положение игрока в модели Player

player = Player(

game\_field=game\_field, # Если Player связан с GameField

x=player\_data.get('x', 0),

y=player\_data.get('y', 0)

)

player.save()

messages.success(request, "Задача успешно сохранена!")

return JsonResponse({'status': 'success'}) # Возвращаем JSON-ответ

else:

messages.error(request, "Заполните все поля!")

return JsonResponse({'status': 'errorrrr', 'errors': fields\_form.errors}, status=400)

def get\_chapters(request):

chapters = Task.objects.values('chapter').distinct()

chapter\_list = [chapter['chapter'] for chapter in chapters]

print(chapter\_list)

return JsonResponse(chapter\_list, safe=False)

def get\_levels(request, chapter\_name):

levels = Task.objects.filter(chapter=chapter\_name).values('level')

task\_id = Task.objects.filter(chapter=chapter\_name).values('id')

task\_id\_list = [task['id'] for task in task\_id]

level\_list = [level['level'] for level in levels]

result = dict(zip(task\_id\_list, level\_list))

print(result)

return JsonResponse(result, safe=False