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, математики и электроники

Факультет информатики

Кафедра программных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
  
 к курсовому проекту по дисциплине «Программная инженерия»

по теме «Автоматизированная обучающая система

по основам алгоритмизации и программирования»

Студент И.Р. Азаматов

Студент С.А. Сулименко

Студент А.Д. Черкасов

Руководитель Л.С. Зеленко

Самара 2021

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, математики и электроники

Факультет информатики

Кафедра программных систем

ЗАДАНИЕ

на курсовой проект по дисциплине

«Программная инженерия»

студентам группы № 6413-020302D

И.Р. Азаматову

С.А. Сулименко

А.Д. Черкасову

1. Тема проекта:«Автоматизированная обучающая система по основам алгоритмизации и программирования»
2. Исходные данные к проекту**:** см. приложение к заданию
3. Перечень вопросов, подлежащих разработке:
   1. Произвести анализ предметной области: изучить основные принципы составления обучающих заданий по основам алгоритмизации и программирования
   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 курсового проекта
   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 | 15.09.2021 |  |
| 2 | Описание и анализ предметной области | 10 | 06.10.2021 |  |
| 3 | Проектирование системы | 40 | 01.12.2021 |  |
| 3.1 | Разработка структурной схемы системы | 5 | 13.10.2021 |  |
| 3.2 | Разработка функциональной спецификации системы и прототипа интерфейса пользователя | 10 | 27.10.2021 |  |
| 3.3 | Разработка информационно-логического проекта системы и его предъявление руководителю | 20 | 01.12.2021 |  |
| 4 | Реализация проекта, разработка контрольных примеров. Предъявление реализации руководителю | 40 | 15.12.2021 |  |
| 5 | Корректировка проекта и оформление документации проекта. Защита проекта с представлением презентации. | 10 | 29.12.2021 |  |

Задание принял  
 к исполнению 08.09.2021 И.Р. Азаматов

08.09.2021 С.А. Сулименко

08.09.2021 А.Д. Черкасов

ПРИЛОЖЕНИЕ  
к заданию на курсовой проект  
студентам группы № 6413-020302D  
И.Р. Азаматову

С.А. Сулименко

А.Д. Черкасову

Тема проекта: «Автоматизированная обучающая система по основам алгоритмизации и программирования»

Исходные данные к проекту:

1. Характеристика объекта автоматизации:

## объект автоматизации: процесс обучения основам алгоритмизации и программирования;

## виды автоматизируемой деятельности:

* + процесс регистрации пользователей;
  + процесс авторизации пользователей;
  + процесс составления обучающего задания;
  + процесс составления алгоритма для решения обучающего задания;
  + процесс работы с обучающимися группами;
  + процесс визуализации работы составленного алгоритма;
  + процесс сбора и визуализации статистики;

## количество ролей пользователей – 2;

## минимальная длина логина – 4 символа;

## максимальная длина логина – 16 символов;

## минимальная длина пароля – 6 символа;

## максимальная длина пароля – 16 символов;

## форма поля – квадрат;

## минимальная длина стороны поля – 4 клеток;

## максимальная длина стороны поля – 12 клеток;

## количество видов клеток – 5;

## минимальное количество корзин на игровом поле – 1;

## максимальное процентное соотношение количества клеток-корзин к количеству всех клеток – 10%;

## количество обезьянок на игровом поле – 1;

## максимальное процентное соотношение количества клеток-бананов к количеству всех клеткам – 30%;

## максимальное процентное соотношение количества клеток-лиан к количеству всех клеток – 20%;

## количество команд, выполняемых мышкой – 6;

## максимальная вложенность циклов – 2;

## максимальное количество итераций цикла – 16;

## максимальное количество символов в текстовом задании – 256;

## минимальное количество баллов за выполненное задание – 0;

1. максимальное количество баллов за выполненное задание – 5.
2. Требования к информационному обеспечению:
3. информационное обеспечение разрабатывается на основе следующих источников:
   * Звенигородский Г. А. Первые уроки программирования/Под редакцией А. П. Ершова. М.: Наука. Главная редакция физико-математической литературы, 1985, 208 с.
4. структура базы данных разрабатывается на основании следующих сведений:
   * о пользователе (логин, пароль, роль, имя, фамилия);
   * о задаче (название, текстовое описание, дата создания, автор задачи, игровая карта);
   * о решении (автор решения, дата решения, список команд, оценка, количество попыток);
5. должна быть обеспечена целостность базы данных и защита от несанкционированного доступа;
6. графические примитивы хранятся в файлах формата \*.png.
7. Требования к техническому обеспечению:
   1. Требования к техническому обеспечению серверной части:
8. тип ЭВМ – IBM PC совместимый;
9. объем ОЗУ – не менее 4 Гб;
10. объем свободного пространства на внешнем диске – не менее 30 Гб;
11. наличие подключения к сети Интернет;
12. технические характеристики определяются в процессе выполнения проекта.
    1. Требования к техническому обеспечению клиентской части:
13. тип ЭВМ – IBM PC совместимый;
14. монитор с разрешающей способностью не ниже 800 х 600;
15. манипулятор – мышь;
16. технические характеристики определяются в процессе выполнения проекта.
17. Требования к программному обеспечению:
    1. Требования к программному обеспечению серверной части:
18. тип операционной системы – UNIX-подобные системы, Windows 7 и выше;
19. СУБД – PostgreSQL 10.
    1. Требования к программному обеспечению клиентской части:
20. тип операционной системы – Windows 7 и выше;
21. браузер – Google Chrome 86.0.4240.183 (64-битный) и выше, Яндекс.Браузер 19.1 и выше.
    1. Требования к программному обеспечению рабочего места разработчика:
22. тип операционной системы – Windows 8 и выше;
23. язык программирования – Java, TypeScript;
24. среда программирования – IntelliJ IDEA 2019;
25. СУБД – PostgresSQL 10;
26. среда проектирования – StarUML 4.0.1.
27. Общие требования к проектируемой системе:

5.1 Функции, реализуемые системой:

1. общесистемные функции:
   * авторизация пользователя в системе (ввод логина, пароля);
   * регистрация пользователя в системе (ввод логина, пароля, фамилия, имя, отчество);
   * аутентификация пользователя в системе, настройка интерфейса пользователя на заданную роль;
   * визуализация процессов работы с игровым полем;
   * выдача справочной информации о системе;
2. функции учителя:
   * создание задачи:
3. ввод названия задачи;
4. создание игрового поля;
5. ввод описания задачи;
   * настройка игрового поля:

* выбор способа создания игрового поля;
* задание длины стороны поля;
* выбор вида клетки;
* расположение клетки на игровом поле;
* задание соотношения типов клеток;
  + сохранение задачи в базу данных;
  + работа со списком учеников:

1. добавление ученика в группу;
2. удаление ученика из группы;
3. проверка и оценивание решений ученика;
   * просмотр статистики по всем ученикам группы;
   * просмотр созданных задач;
4. функции ученика:
   * загрузка задачи из базы данных;
   * составление алгоритма решения задачи:
5. добавление команды;
6. удаление команды;
7. перемещение команды;
8. настройка количества итераций цикла;
   * запуск созданного алгоритма решения задачи;
   * сохранение алгоритма решения задачи;
   * отправка задания на проверку;
   * просмотр оценок;

5.2 Технические требования к системе:

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

Руководитель   
проекта 06.10.2021 Л.С. Зеленко

Задание принял  
 к исполнению 08.09.2021 И.Р. Азаматов

08.09.2021 С.А. Сулименко

08.09.2021 А.Д. Черкасов

РЕФЕРАТ

Пояснительная записка 40 с, 14 рисунков, 5 таблиц[[1]](#footnote-1), 12 источников,  
2 приложения.

Графическая часть: ??? слайдов презентации PowerPoint.

ДЕРЕВО ПОИСКА, ГЕНЕРАТОР КРОССВОРДОВ, ГОЛОВОЛОМКА, СЛОВАРЬ ТЕРМИНОВ, ВАРИАНТ ОТОБРАЖЕНИЯ, РАЗГАДЫВАНИЕ

Во время курсового проектирования разработаны алгоритмы и соответствующая им программа, позволяющая выполнять автоматическую генерацию линейного кроссворда по заданной теме. Задания (понятие и его расшифровка) хранятся в текстовом файле и могут дополняться вручную внутри программы, при этом ограничений на длину словаря не существует. Тема кроссворда выбирается пользователем в соответствии с содержанием словаря заданий. Программа позволяет сформировать кроссворд, учитывая ограничения на параметры. В системе имеется возможность сохранения кроссвордов в файл с целью последующего их разгадывания.

Программа написана на языке С# в среде Visual Studio 2015 и функционирует под управлением операционной системы Windows 7 и выше. Доступ к данным осуществляется с помощью СУБД PostgrateSQL 10.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 13](#_Toc86217833)

[1 Описание и анализ предметной области 15](#_Toc86217834)

[1.1 Описание предметной области 15](#_Toc86217835)

[1.1.1 Автоматизированная обучающая система: определение, классификация, применение 15](#_Toc86217836)

[1.1.2 Обучение основам алгоритмизации и программирования 17](#_Toc86217837)

[1.1.3 Задача Звенигородского Г. А. про муравья 18](#_Toc86217838)

[1.2 Описание систем-аналогов 20](#_Toc86217839)

[1.2.4 Обучающая система «Scratch» 20](#_Toc86217840)

[1.2.5 Платформа «Codewars» 22](#_Toc86217841)

[1.3 Диаграмма объектов предметной области 24](#_Toc86217842)

[1.4 Постановка задачи 25](#_Toc86217843)

[2 Проектирование системы 30](#_Toc86217844)

[2.1 Выбор и обоснование архитектуры системы 30](#_Toc86217845)

[2.1.1 Клиент-серверная архитектура 32](#_Toc86217846)

[2.1.2 «Тонкий» клиент 34](#_Toc86217847)

[2.1.3 «Толстый» клиент 34](#_Toc86217848)

[2.1.4 Выбор протокола обмена данными 34](#_Toc86217849)

[2.2 Структурная схема системы 35](#_Toc86217850)

[2.3 Разработка спецификации требований 38](#_Toc86217851)

[2.3.1 Функциональная спецификация 40](#_Toc86217852)

[2.3.2 Перечень исключительных ситуаций 40](#_Toc86217853)

[2.4 Разработка прототипа интерфейса пользователя системы 53](#_Toc86217854)

[2.5 Разработка информационно-логического проекта системы 59](#_Toc86217855)

[2.5.3 Язык UML 59](#_Toc86217856)

[2.5.4 Диаграмма вариантов использования 59](#_Toc86217857)

[2.5.5 Сценарии 62](#_Toc86217858)

[2.5.6 Диаграмма классов 62](#_Toc86217859)

[2.5.7 Диаграмма состояний 63](#_Toc86217860)

[2.5.8 Диаграмма деятельности 63](#_Toc86217861)

[2.5.9 Диаграмма последовательности 63](#_Toc86217862)

[2.6 Логическая модель данных (при необходимости) 64](#_Toc86217863)

[2.7 Выбор и обоснование алгоритмов обработки данных /Разработка и описание алгоритмов обработки данных 66](#_Toc86217864)

[2.8 Выбор и обоснование комплекса программных средств 66](#_Toc86217865)

[2.8.1 Выбор языка программирования 67](#_Toc86217866)

[2.8.2 Выбор операционной системы 67](#_Toc86217867)

[2.8.3 Выбор среды программирования 67](#_Toc86217868)

[2.8.4 Выбор системы управления базами данных (при необходимости) 67](#_Toc86217869)

[3 Реализация системы 68](#_Toc86217870)

[3.1 Разработка и описание интерфейса пользователя 68](#_Toc86217871)

[3.1.1 Разработка и описание пользовательского меню 68](#_Toc86217872)

[3.1.2 Описание тестового примера 68](#_Toc86217873)

[3.2 Диаграммы реализации 68](#_Toc86217874)

[3.2.3 Диаграмма компонентов 68](#_Toc86217875)

[3.2.4 Диаграмма развертывания 68](#_Toc86217876)

[3.2.5 Диаграмма классов 69](#_Toc86217877)

[3.3 Физическая модель данных (при необходимости) 69](#_Toc86217878)

[3.4 Выбор и обоснование комплекса технических средств 70](#_Toc86217879)

[3.4.1 Расчет объема занимаемой памяти 70](#_Toc86217880)

[3.4.2 Минимальные требования, предъявляемые к системе 72](#_Toc86217881)

[ЗАКЛЮЧЕНИЕ 73](#_Toc86217882)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 74](#_Toc86217883)

[ПРИЛОЖЕНИЕ А Руководство пользователя 77](#_Toc86217884)

[А.1 Назначение системы 77](#_Toc86217885)

[А.2 Условия работы системы 77](#_Toc86217886)

[А.3 Установка системы 77](#_Toc86217887)

[А.4 Работа с системой 78](#_Toc86217888)

[А.4.1 Работа с системой в режиме администратора (если необходимо) 78](#_Toc86217889)

[А.4.2 Работа с системой в режиме пользователя 78](#_Toc86217890)

[ПРИЛОЖЕНИЕ Б Листинг модулей программы 79](#_Toc86217891)

ВВЕДЕНИЕ

С приходом эпохи информационных технологий и автоматизации мир начал очень быстро меняться. Многих работников заменили машины, а получать знания стало намного проще чем раньше. Не могла остаться без внимания и система образования, в списке школьных предметов появилась информатика. В 1984 году вышло постановление о проведении школьной реформы, в которой среди пяти важнейших направлений повышения качества учебно-воспитательного процесса было сказано, что учащихся необходимо «вооружать» знаниями и навыками использования современной вычислительной техники, а также обеспечить широкое применение компьютеров в учебном процессе [1].

Умение работать с электронно-вычислительными машинами и информацией становится необходимыми для каждого человека, ведь всё больше услуг можно получить в электронном виде через специальные сервисы. Специалисты, разрабатывающие такие приложения, обладают навыками программирования и алгоритмизации, а также высоко ценятся на рынке труда. В связи с этим появляется актуальность в создании обучающих автоматизированных систем, благодаря которым человек сможет получить данные навыки.

В настоящее время уже существует большое количество обучающих автоматизированных систем, в том числе основам алгоритмизации и программирования для людей всех возрастов. Они обладают различными функциями и возможностями для обучения.

Во время курсового проектирования необходимо разработать автоматизированную систему для обучения основам алгоритмизации и программирования, помощью которой преподаватель сможет составлять обучающие задания и следить за успеваемостью своих учеников, которые в свою очередь будут решать поставленные задачи при помощи графических примитивов.

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 [2]. При проектировании системы будут использоваться методология ООАП (Object-Oriented Analysis/Design), в основу которой положена объектно-ориентированная методология представления предметной области в виде объектов, являющихся экземплярами соответствующих классов, и язык моделирования UML (Unified Modeling Language), который является стандартным инструментом для разработки «чертежей» программного обеспечения [3].

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» [4].

* 1. Описание предметной области
     1. Автоматизированная обучающая система: определение, классификация, применение

Автоматизированная обучающая система (АОС) – взаимно согласованная совокупность учебных материалов, средств их разработки, хранения, передачи и доступа к ним, предназначенная для целей обучения и основания на использовании современных информационных технологий [5].

Существующие обучающие системы делят на два класса [6]:

1. обучающие системы, в которых управление процессом обучения возложено на пользователя;
2. обучающие системы, самостоятельно управляющие учебным процессом.

Первый класс содержит изложение учебной дисциплины или ее раздела в соответствии с ее логикой на машинном носителе в текстовом и графическом форматах. Обучающие системы данного класса отличаются между собой функциональностью, свойствами, способами их реализации. Например, такую обучающую систему, как электронный учебник можно рассматривать как электронную копию традиционного печатного учебника или пособия. Структура представления материала на машинном носителе является последовательной или имеет гипертекстовую структуру.

Такая обучающая система, как электронная библиотека, является системой, управляющей комплексом электронных учебно-методических материалов различного класса по различным учебным дисциплинам, позволяющей обучаемому выполнять поиск информации (поиск, по ключевым словам, поиск по предметной области). Мультимедиа является электронным учебником, в котором изложение учебной дисциплины полностью выполнено или дополнено изложением в аудио, видео форматах. Данная система позволяет обучаемому наблюдать динамику изучаемых явлений и изменять параметры этой динамики [7].

Второй класс содержит изложение учебной дисциплины или ее раздела в соответствии с ее логикой на машинном носителе в текстовом, графическом, аудио, видео форматах. В конце каждой порции изложения учебной дисциплины в данных системах обучаемому предоставляются проверочные задания. В отличие от систем первого класса, в данных системах ответы и действия обучаемого влияют на дальнейший ход процесса обучения. Степень управления учебным процессом напрямую зависит от степени адаптации системы конкретного обучаемого.

Некоторые обучающие системы обладают функциями контроля знаний. Эти функции позволяют сделать более эффективным контроль, а также эффективным становится и сам учебный процесс. При помощи контроля определяется исходный уровень для дальнейшего овладения знаниями, умениями и навыками, изучается глубина и объем их усвоения.

Тренажерная обучающая система позволяет дополнить полученные в учебно-справочной системе знания полной отработкой понятийных логических и моторных навыков работы с технологическими объектами [6].

Применение обучающих систем позволяет [7]:

* решить проблемы с одновременным обучением значительного количества пользователей;
* решить проблемы с обучением географически распределенных пользователей;
* обеспечить обучение пользователей в удобное для них время;
* повысить конкурентоспособность за счет предоставления клиентам возможности дистанционного обучения правилам использования продуктов и услуг;
* существенно сократить затраты на обновление учебного материала (нет необходимости актуализировать информацию у каждого пользователя, когда можно один раз обновить информацию учебных модулей (курсов), размещенных на сервере);
* представить информацию в максимально наглядной форме за счет возможности использования в составе учебных курсов различного медиа-контента.
  + 1. Обучение основам алгоритмизации и программирования

Вопросы обучения детей основам алгоритмизации и программирования в научной и научно-методической литературе обсуждаются учеными, учителями и специалистами в области информационных технологий. Рассматривая программирование, как общение с компьютером на языке понятном ему, исследователи поддерживают идею обучения детей программированию с раннего возраста [8].

Однако, при обучении программированию в раннем возрасте возникает следующая проблема: учащиеся начальных классов не могут запоминать сложные команды, длинные коды, написанные, как правило, на иностранном языке. Для решения данной проблемы необходим язык программирования близкий к образу мышления детей, содержащий команды для работы с интересными и понятными для них объектами, но в то же время, дающий прочную основу для изучения других языков программирования.

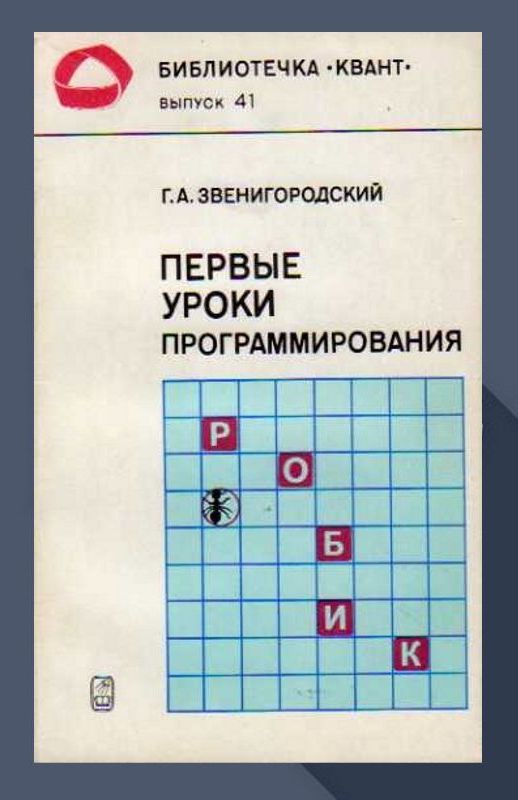
В разработке одного из такого языка принимал участие один из основателей советской школьной (учебной) информатики – Звенигородский Г. А. Он занимался созданием и поддержкой языка начального обучения программированию Робик и более мощного учебно-производственного языка РАПИРА, который использовался при разработке систем школьной информатики. Им же вместе с его учениками была создана и система программирования для этих языков «Школьница», а также несколько других вспомогательных языков (например, графическая система «ШПАГА» – Школьный ПАкет Графических процедур, Адаптированный) [9].

Однако на этом вклад Звенигородского Г. А. в обучение детей основам алгоритмизации и программирования не ограничивается. Г. А. Звенигородский был одним из организаторов Летних школ юных программистов, которые проводятся в новосибирском Академгородке с 1976 г. до сих пор, Всесоюзной заочной школы программирования, организованной на базе журнала для школьников «Квант». Он регулярно вёл занятия в воскресной школе юных программистов при ВЦ СО АН СССР.

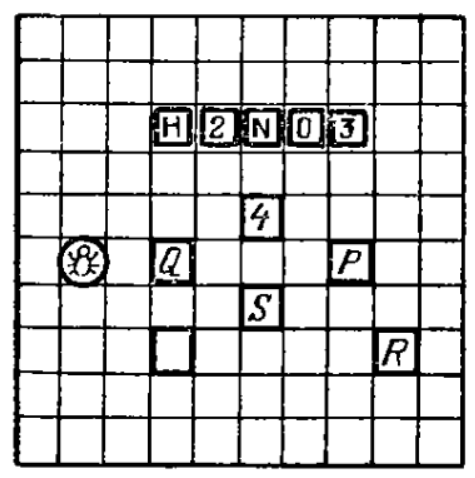
Ещё одним достижением Звенигородского Г. А. в этой области является выпуск книги «Первые уроки программирования» [10] в 1985 году (рисунок 1). Цель книги – способствовать формированию у школьников умений и навыков, необходимых для взаимодействия с ЭВМ на языках высокого уровня. В ней даются конструкции таких языков и основные приемы прикладного программирования.

* + 1. Задача Звенигородского Г. А. про муравья

В качестве основы для разрабатываемой системы обучения алгоритмизации и программирования возьмем знаменитую задачу Звенигородского Г. А. про муравья, описанную в книге «Первые уроки программирования». Исполнитель в данной задаче – муравей, он может ползать по доске (обычно 10 на 10 клеток) и передвигать по ней кубики с надписями или рисунками (рисунок 2). За один «ход» муравей сдвигается на одну клетку вверх, вниз, вправо или влево (но не по диагонали!). Если он натыкается на кубик или на несколько кубиков, он будет толкать их перед собой, пока не изменит направление движения или не упрется в край доски.

  
Рисунок 1 – Книга Звенигородского Г. А. «Первые уроки программирования»

Считается, что края доски ограждены бортиками; попытка толкать кубики дальше вызывает сообщение об ошибке. За движениями исполнителя можно следить в интерактивном режиме на экране, подключенного к вычислительной машине.

  
Рисунок 2 – Игровая доска в задаче Звенигородского Г. А. про муравья

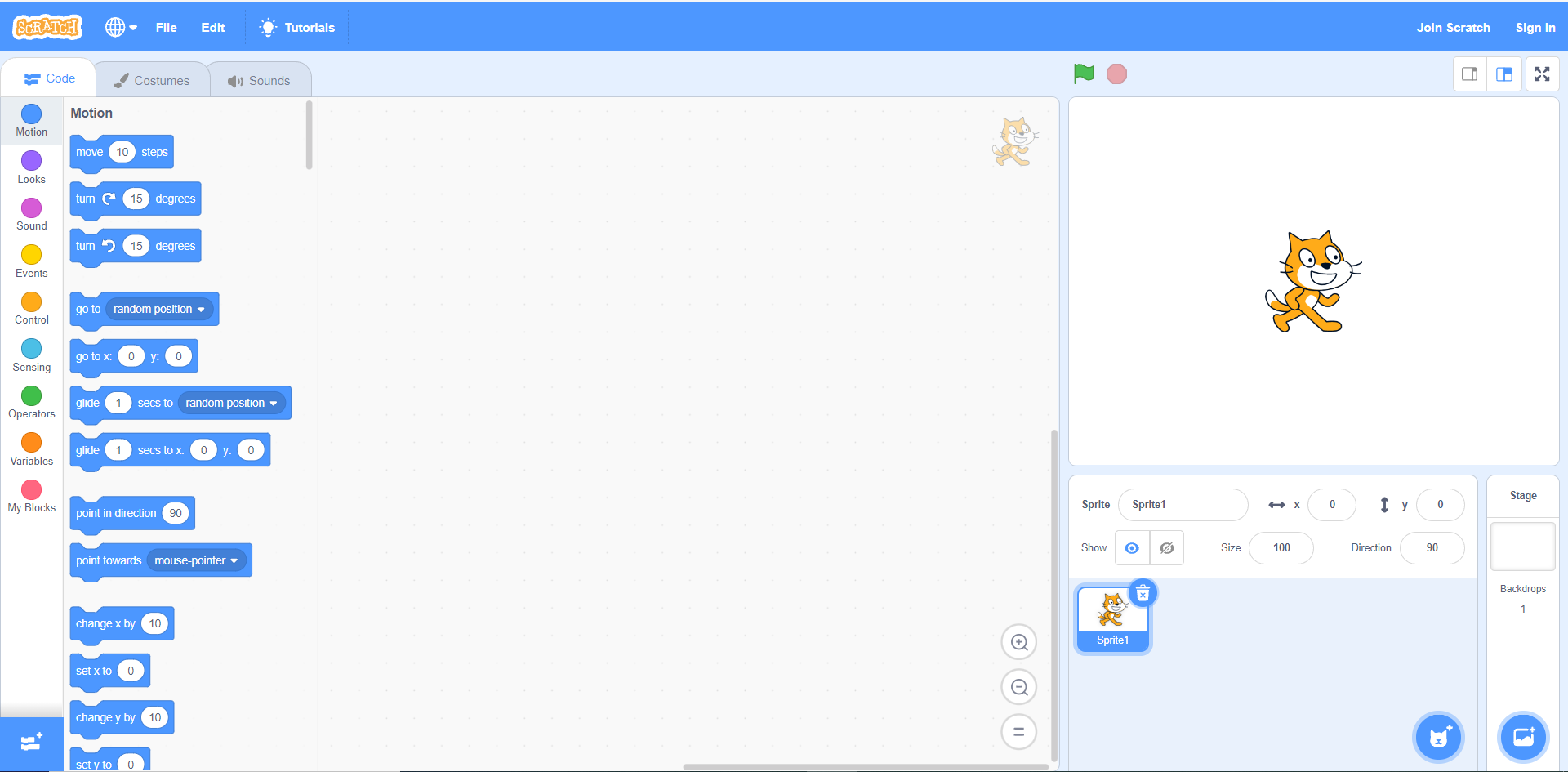
* 1. Описание систем-аналогов

На сегодняшний день существует несколько ресурсов, направленных на обучение основам алгоритмизации и программирования. Каждый из таких ресурсов имеет свои особенности и функции, которые будут рассмотрены в следующих пунктах.

* + 1. Обучающая система «Scratch»

Scratch – это визуальная среда программирования, где в отличие от других языков программирования, необходимо собирать код программы из блоков. С помощью Scratch можно программировать собственные интерактивные истории, игры и мультфильмы, а затем делиться результатами с другими участниками онлайн-сообщества. Scratch помогает молодым людям учиться мыслить творчески, рассуждать системно и работать совместно – это необходимые навыки для жизни в 21 веке [11].

На рисунке 3 приведено главное окно среды разработки Scratch, которое разделено на три секции:

  
Рисунок 3 – Главное окно среды разработки «Scratch»

2

3

1

1. секция с блоками кода;
2. секция с полем для создания кода;
3. секция с отображением результата выполнения программы;

Секция с блоками кода представляет собой меню, с различными типами блоков:

* блоки передвижения (motion);
* блоки изменения визуального отображения (looks);
* блоки звуков (sound);
* блоки событий (events);
* блоки с циклами и операторами ветвления (control);
* блоки взаимодействия с пользователем (sensing);
* блоки операторов (operators);
* блоки переменных (variables);
* блоки, которые может создавать сам пользователь (my blocks).

При выборе любого блока, справа отображается палитра элементов, содержащихся в этом блоке. Каждый элемент представляет собой карточку, с различными командами и полями, для настройки программы.

В секцию поля элементы попадают путем захвата и перемещения мышью нужного элемента. После размещения элементы можно настраивать – изменять значения переменных, выбирать доступные действия. При установке 2 элементов рядом образуется скрипт, который будет выполняться при наступлении одного из событий из блока событий. Каждый скрипт ассоциируется со своим спрайтом. Спрайт – это объект в стиле Scratch, ассоциированный с изображением и набором переменных и скриптов, которые определяют его поведение. У каждого спрайта может быть несколько скриптов, ассоциирующихся с разными событиями.

Выполнение программы визуализируется в 3 секции. Она представляет собой сцену, на которой визуализируется скрипт, а также содержит набор спрайтов, которые отображаются на сцене.

Данный сайт позволяет освоить азы сразу 3 парадигм программирования [12]:

## Структурная (в низкоуровневом понимании) − все программы конструируются из ограниченного набора элементов (блоков).

## Объектно-ориентированная − каждый спрайт на самом деле является объектом со своими свойствами (переменными) и поведением (скриптами), и разные объекты могут взаимодействовать.

1. Многопоточная − объекты взаимодействуют посредством обмена сообщениями через блоки «broadcast» и «when I receive».

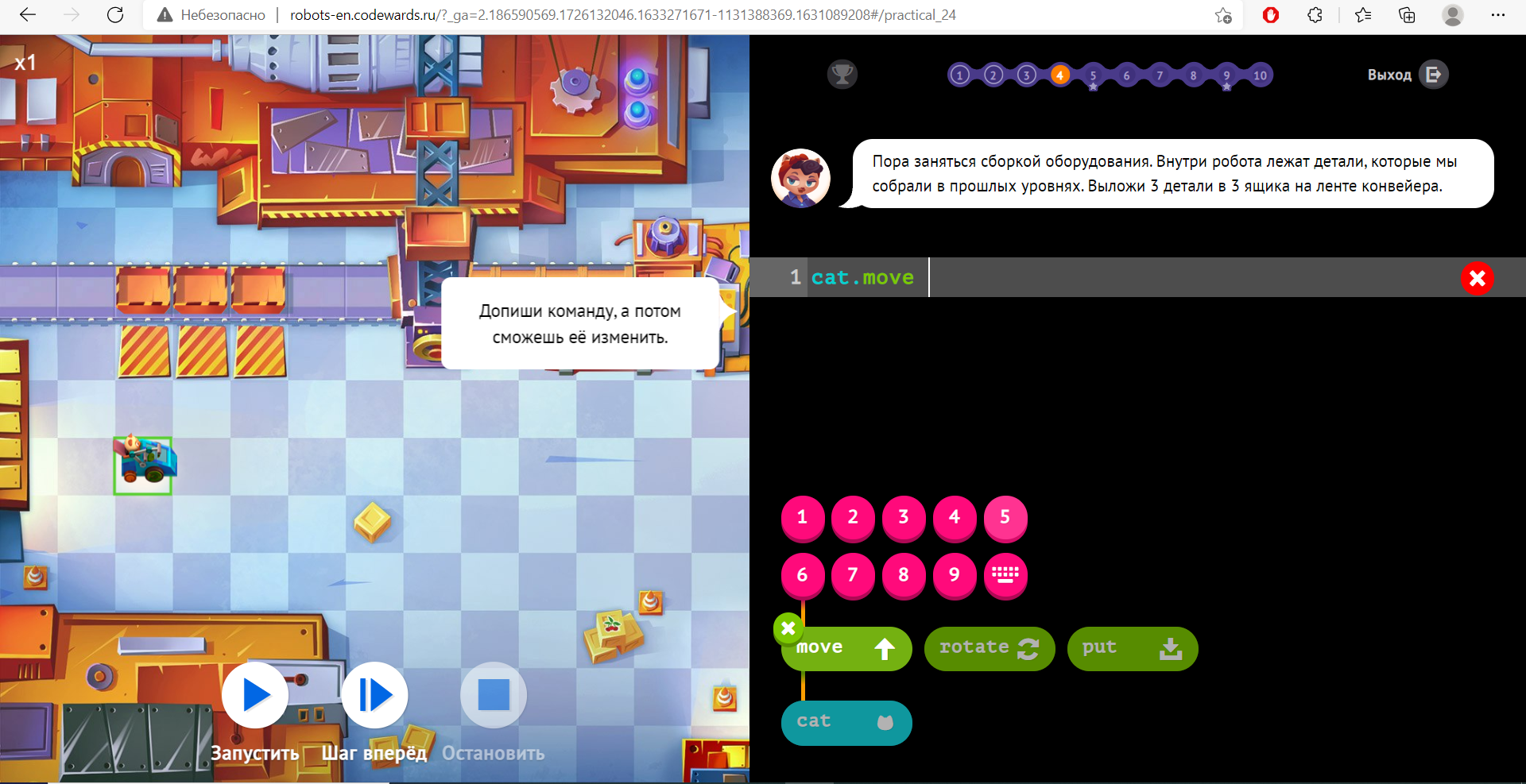
К достоинствам данной системы относятся:

* качественная визуализация;
* большое количество методических пособий и учебных материалов;
* удобный интерфейс пользователя;
* бесплатность ресурса.

К недостаткам системы относятся:

* неудобство проверки написанного кода;
* отсутствие автоматической проверки заданий.
  + 1. Платформа «Codewars»

Codewars – это платформа для обучения детей основам программирования. Она предоставляет собой среду для выполнения заданий при помощи графического программирования [13]. На рисунке 4 приведена страница выполнения задания «Codewars».

  
Рисунок 4 – Основной экран платформы «Codewars»

2

1

Страница выполнения задания делится на 2 секции:

1. секция поля;
2. секция кода программы.

В секции поля расположено само поле, представляющее собой сетку квадратов, с различными препятствиями. Также там расположены кнопки для управления выполнением скрипта.

В секции кода имеется счетчик количества заданий. Далее расположен блок с самим заданием. Ниже, под заданием, расположено тело скрипта, который можно заполнять как с клавиатуры, так и при помощи графического интерфейса. Доступные блоки расположены под скриптом. При нажатии на них скрипт дополняется соответствующими командами. Все команды представляют собой иерархическую структуру.

Выполнение задания проверяется автоматически.

Данный портал позволяет развивать следующие навыки:

* ввод текста с клавиатуры;
* оперирование числами, начальные математические знания;
* проведение простейших измерений разными способами;
* наглядное представление данных;
* различение частей одного объекта, сравнение и понимание различий по размерам, расположению в пространстве;
* видение рисунка решения, понимание структуры кода;
* составление алгоритмов, с применением оптимальных концепций.

К достоинствам данной системы относятся:

* качественная визуализация;
* удобный интерфейс;
* автоматическая проверка решения.

К недостаткам системы относятся:

* отсутствует возможность составления своих заданий;
* платформа является платной.

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

Таблица 1 – Сравнительные характеристики систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название показателя | «Scratch» | «Codewars» | Разрабатываемая система |
| Возможность составления собственных заданий | Присутствует | Отсутствует | Присутствует |
| Автоматическая генерация задания | Отсутствует | Присутствует | Присутствует |
| Возможность ручной проверки решения | Присутствует | Присутствует | Присутствует |
| Автоматическая проверки решения | Отсутствует | Присутствует | Отсутствует |
| Различные режимы для учителя и ученика | Отсутствуют | Присутствуют | Присутствуют |
| Бесплатность платформы | Да | Нет | Да |
| Качественная визуализация | Да | Да | Да |
| Удобный интерфейс | Да | Да | Да |

* 1. Диаграмма объектов предметной области

Для моделирования предметной области используется объектно-ориентированное проектирование. Объектно-ориентированное проектирование – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы [14].

Наиболее наглядно объекты предметной области могут быть изображены с помощью диаграммы. Для этих целей отличным инструментом служат UML диаграммы. UML – это унифицированный язык моделирования, предназначенный для создание моделей, описывающих какую-либо предметную область и подходящий для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. [15]

На рисунке 5 приведена диаграмма объектов предметной области. Имеется 2 участника: учитель и ученик.

Одним из основных объектов является учебное задание, которое состоит из игрового поля и описания задания. Оно составляется учителем. Игровое поле в свою очередь состоит из клеток: лиана, пустая клетка, корзина, банан, обезьянка – объект, перемещающийся по игровому полю. Сама клетка – это графический примитив, который хранится в базе данных.

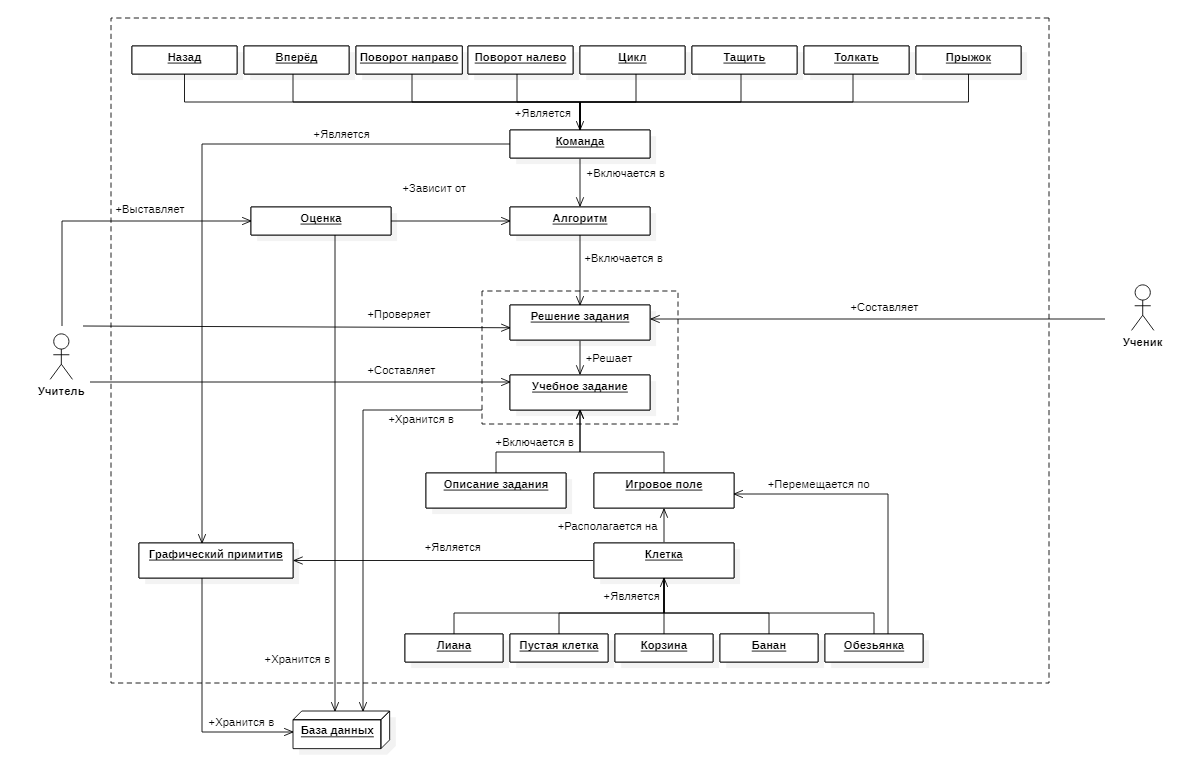
Еще один важный объект – это решение задания, которое решает поставленную учителем задачу. Решение задания составляет ученик, и проверяет учитель. Оно включает в себя алгоритм, за который учитель ставит оценку, которая сохраняется в базе данных. Алгоритм – это набор команд, которым так же соответствует графический примитив, хранящийся в базе данных. Команды могут быть следующие: назад, вперед, поворот направо, поворот налево, цикл, прыжок.

Решение задания и само учебное задание также хранятся в базе данных.

* 1. Постановка задачи

Во время курсового проектирования необходимо разработать автоматизированную обучающую систему основам алгоритмизации и программирования. Система должна быть построена на клиент-серверной архитектуре и представлять собой веб-приложение. Все необходимые данные для работы системы будут храниться в базе данных на сервере. Система позволит учителям выдавать задания ученикам, которые будут их решать, а также проверять решенные учениками задания и просматривать статистику.

В системе должны быть присутствовать две роли: учитель и ученик.

  
Рисунок 5 – Диаграмма объектов предметной области

Учитель и ученик должны иметь возможность зарегистрироваться, введя логин (от 4 до 16 символов), пароль (от 6 до 16 символов), фамилию, имя и отчество, а также выбрать роль (учитель или ученик). После регистрации, учитель и ученик смогут авторизоваться на сайте по логину и паролю, при этом, интерфейс приложения будет автоматически настроен на необходимую роль. И для ученика, и для преподавателя будет предусмотрена возможность визуализации процессов работы с игровым полем, а также они смогут просматривать справочную информацию о системе.

Режим «Учитель»

В данном режиме учителю будет предоставлена возможность создания задачи, которая включает в себя ввод ее названия и описания (не более 256 символов), а также создание квадратного игрового поля со стороной от 4 до 12 клеток. После создания макета игрового поля, учитель сможет настроить игровое поле автоматически, либо вручную. При ручном заполнении поля, учитель будет определять тип для клеток (пустая клетка, лиана, корзина, банан или обезьянка и их расположение на игровом поле. После заполнения поля будет произведена проверка его корректности, которая определяется в зависимости от определенных параметров, таких как количество обезьянок на игровом поле (одна), количество корзин (не менее одной), максимальное процентное соотношение количества клеток-корзин к количеству всех клеток (не более 10%), максимальное процентное соотношение количества клеток-бананов к количеству всех клеток (не более 30%) и максимальное процентное соотношение количества клеток-лиан к количеству всех клеток (не более 20%). При автоматической генерации поля учителю нужно будет выставить только процентное соотношение перечисленных выше параметров. Созданную задачу учитель сможет сохранить в базу данных. Кроме перечисленных функций, учитель будет иметь возможность добавлять к себе в группу учеников, проверять решенные ими задания и выставлять за них баллы (от 0 до 5), а также просматривать статистику по ученикам.

Режим «Ученик»

В данном режиме ученику будет доступна возможность решения задач, которые составил учитель. Ученику будет необходимо составить алгоритм, используя графические примитивы (поворот влево, поворот вправо, вперёд, назад, прыжок, цикл), на которые будут налагаться ограничения (максимальная вложенность циклов не должна превышать 2, а количество итераций – не более 16). После составления алгоритма ученик имеет возможность запустить его, сохранить и отправить учителю на проверку. Также ученик будет иметь возможность просмотреть оценки за решенные ими задания.

Таким образом, системы должна решать следующие задачи:

1. общесистемные функции:
   * авторизация пользователя в системе (ввод логина, пароля);
   * регистрация пользователя в системе (ввод логина, пароля, фамилия, имя, отчество);
   * аутентификация пользователя в системе, настройка интерфейса пользователя на заданную роль;
   * визуализация процессов работы с игровым полем;
   * выдача справочной информации о системе;
2. функции учителя:
   * создание задачи:
3. ввод названия задачи;
4. создание игрового поля;
5. ввод описания задачи;
   * настройка игрового поля:

* выбор способа создания игрового поля;
* задание длины стороны поля;
* выбор вида клетки;
* расположение клетки на игровом поле;
* задание соотношения типов клеток;
  + сохранение задачи в базу данных;
  + работа со списком учеников:

1. добавление ученика в группу;
2. удаление ученика из группы;
3. проверка и оценивание решений ученика;
   * просмотр статистики по всем ученикам группы;
   * просмотр созданных задач;
4. функции ученика:
   * загрузка задачи из базы данных;
   * составление алгоритма решения задачи:
5. добавление действия;
6. удаление действия;
7. перемещение действия;
8. настройка количества итераций цикла;
   * запуск созданного алгоритма решения задачи;
   * сохранение алгоритма решения задачи;
   * отправка задания на проверку;
   * просмотр оценок;
9. Проектирование системы
   1. Выбор и обоснование архитектуры системы

Архитектура программного обеспечения – это структура программы или вычислительной системы, которая включает программные компоненты, видимые снаружи свойства этих компонентов, а также отношения между ними.

Существуют следующие виды архитектур программной системы [16]:

1. локальная;
2. файл-серверная;
3. клиент-серверная.

При локальной архитектуре все компоненты программной системы находятся на одном компьютере. Такой вид архитектуры широко использовался до появления компьютерных сетей. Основной недостаток таких систем заключается в том, что работать в программной системе может только один пользователь. Другие пользователи не имеют возможности получить доступ к данным.

При файл-серверной архитектуре, сервер только извлекает данные из файла (файлов) базы данных и передает их клиенту для дальнейшей обработки. В процессе работы из базы данных клиенту передаются большие объемы информации. В файл-серверной архитектуре всегда передаются избыточные данные. Неважно, сколько записей из базы данных нужны клиенту – файлы базы данных передаются в самом общем случае целиком. На рисунке 6 представлена схема файл-серверной архитектуры.

Информационные системы с клиент-серверной архитектурой позволяют избежать проблем файл-серверных приложений. При такой архитектуре сервер базы данных, расположенный на компьютере-сервере, обеспечивает выполнение основного объема обработки данных (см. рисунок 7). Клиентское приложение формирует запросы к серверу базы данных, как правило, в виде инструкций языка SQL.

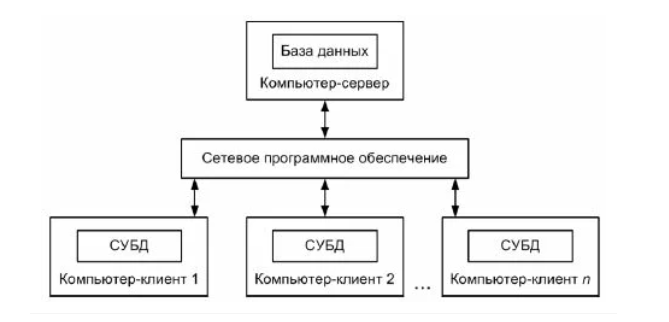
  
Рисунок 6 – Схема файл-серверной архитектуры

  
Рисунок 7 – Схема клиент-серверной архитектуры

Сервер извлекает из базы запрошенные данные и передает на компьютер клиента. Главное достоинство такого подхода − значительно меньший объем передаваемых данных. Клиент-серверная архитектура позволяет разгрузить сеть и поддерживать непротиворечивость данных за счет их централизованной обработки. В настоящее время архитектура «клиент-сервер» широко признана и находит применение для организации работы приложений как для рабочих групп, так и для информационных систем масштаба предприятия [17].

После изучения существующих архитектур была выбрана клиент-серверная архитектура. Это необходимо для того, чтобы ученики могли обучаться независимо друг от друга (одновременно), не создавая высокой нагрузки на сервер.

* + 1. Клиент-серверная архитектура

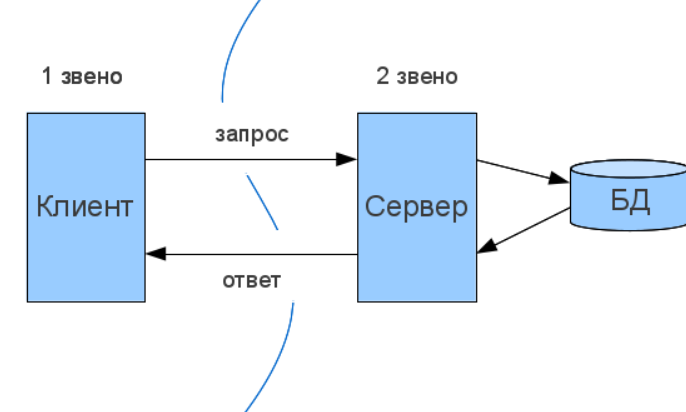
Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты, потребители этих функций [18]. Существуют двухзвенная и трехзвенная клиент-серверная архитектуры.

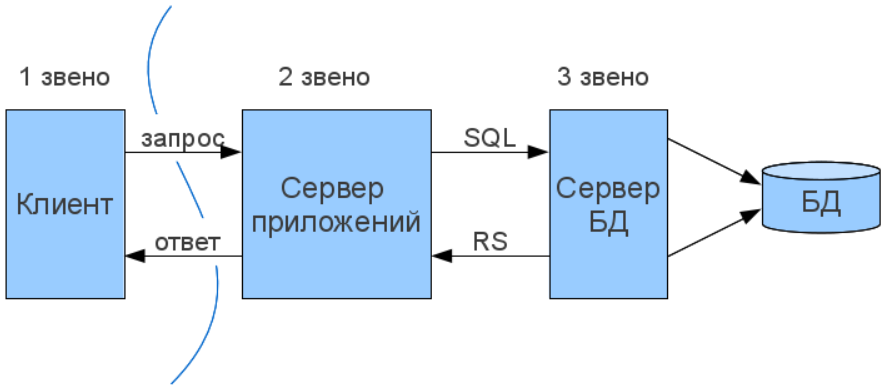
Двухзвенная архитектура

В любой сети, построенной на современных сетевых технологиях, присутствуют элементы клиент-серверного взаимодействия, чаще всего на основе двухзвенной архитектуры. Двухзвенной она называется из-за необходимости распределения трех базовых компонентов между двумя узлами (клиентом и сервером). Двухзвенная архитектура используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме, при этом используя только собственные ресурсы, т.е. сервер не вызывает сторонние сетевые приложения и не обращается к сторонним ресурсам для выполнения какой-либо части запроса [19]. На рисунке 8 приведена схема двухзвенной архитектуры.

Трехзвенная архитектура

Тенденция в клиент-серверных технологиях связана со все большим использованием распределенных вычислений. Они реализуются на 32 основе модели сервера приложений, где сетевое приложение разделено на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная клиент-серверная архитектура становится трехзвенной [20]. На рисунке 9 приведена схема трехзвенной архитектуры.

  
Рисунок 8 – Схема двухзвенной архитектуры

  
Рисунок 9 – Схема трехзвенной архитектуры

Трехзвенная архитектура сложнее, но благодаря тому, что функции распределены между серверами второго и третьего уровня, эта архитектура представляет:

* высокую степень гибкости и масштабируемости;
* высокую безопасность (т.к. защиту можно определить для каждого сервиса или уровня);
* высокую производительность (т.к. задачи распределены между серверами);
* клиент-серверные технологии.

При реализации системы будет использоваться двухзвенная клиент-серверная архитектура.

* + 1. «Тонкий» клиент

Для доступа пользователей к сетевым сервисам необходимы клиенты, но клиенты могут иметь различные возможности, они характеризуются «толщиной». Она определяет конфигурацию оборудования и программное обеспечение, находящееся в распоряжении клиента.

«Тонкий» клиент определяет клиент, вычислительных ресурсов которого достаточно для запуска необходимого сетевого приложения через веб-интерфейс. В таком типе клиента вся или почти вся логика выполняется на сервере. Примером такого клиента служит компьютер с браузером, работающий с веб-приложениями [21].

* + 1. «Толстый» клиент

«Толстым» клиентом является клиент, предоставляющий расширенный функционал, который слабо зависит от сервера. Сервера в таких клиентах чаще всего исполняют роль хранилища данных или же предоставляют дополнительные услуги клиенту. Такое приложение совмещает компонент представления данных (графический пользовательский интерфейс ОС) и прикладной компонент (вычислительные мощности клиентского компьютера). Для обмена данными в системе будет использоваться принцип «толстого» клиента [21].

* + 1. Выбор протокола обмена данными

Для передачи данных между сервером и клиентом могут использоваться различные протоколы. Протокол – это набор соглашений, который определяет обмен данными между различными программами. Протоколы задают способы передачи сообщений и обработки ошибок в сети, а также позволяют разрабатывать стандарты, не привязанные к конкретной аппаратной платформе [22].

Модель TCP/IP – это сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю и предоставляет 4 уровня абстракции, каждый из которых описывается протоколом передачи. Основой данной модели являются протоколы TCP и IP, на которых базируется интернет [22].

Протокол TCP (Transmission Control Protocol), отвечающий за транспортировку данных, разбивает передаваемую информацию на порции и нумерует все порции. С помощью протокола IP (Internet Protocol), который отвечает за определение пути передачи данных, все части передаются получателю. Далее с помощью протокола TCP проверяется, все ли части получены. При получении всех порций TCP располагает их в нужном порядке и собирает в единое целое. Эти протоколы являются базовыми в сети Интернет, поэтому выбрать другие на транспортном и сетевом уровне не представляется возможным. Выбор может быть осуществлен только на прикладном уровне. Рассмотрим наиболее популярные протоколы прикладного уровня.

HTTP (Hyper Text Transfer Protocol) – протокол прикладного уровня передачи данных, изначально – в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных. Является самым популярным протоколом прикладного уровня, а также используется для транспортировки в других протоколах этого уровня. Основой данного протокола является клиент – серверная архитектура [23].

Для передачи данных между серверной и клиентской частью в системе будет использоваться HTTP протокол в связи его нацеленностью на клиент серверное взаимодействие и популярностью в сети Интернет.

* 1. Структурная схема системы

При создании автоматизированных систем, разработчики сталкиваются с главной проблемой – сложностью разрабатываемой системы. Невозможно сразу же на этапе проектирования полностью понимать всю систему в целом. Самый популярный подход в решении этой проблемы – декомпозиция системы на небольшое количество подсистем, которые, в свою очередь, также декомпозируются на подсистемы. Так подход называется «разделяй и властвуй». Формализуя основные правила декомпозиции, получим следующие принципы данного подхода [25]:

* количество связей между отдельными подсистемами должно быть минимальным;
* связность отдельных частей внутри каждой подсистемы должна быть максимальной;
* каждая система должна инкапсулировать своё содержимое;
* каждая подсистема должна иметь чётко определённый интерфейс взаимодействия с другими подсистемами.

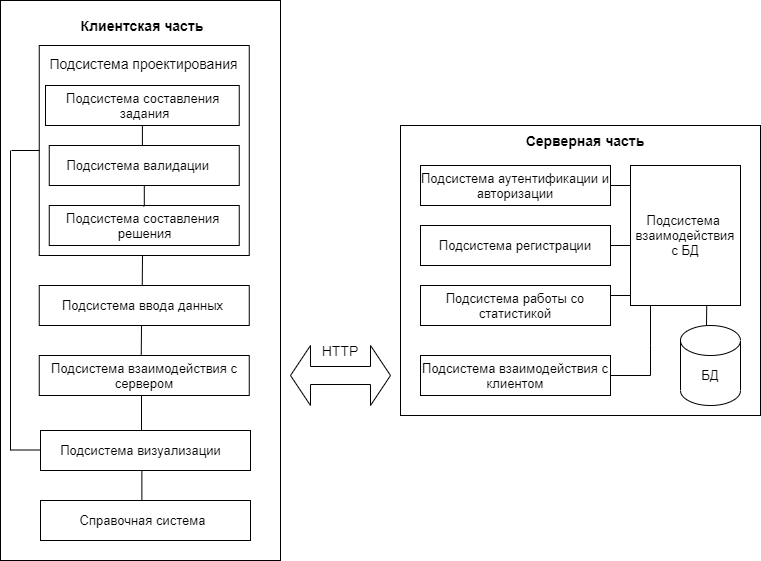
Итак, сущность структурного подхода к разработке автоматизированных систем заключается в её декомпозиции на автоматизируемые функции: система разбивается на функциональные подсистемы, которые, в свою очередь, делятся на подфункции, те — на задачи и так далее до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоcвязаны [25].

На рисунке 10 приведена структурная схема разрабатываемой системы, в ее состав входят клиентская и серверная части, которые взаимодействуют между собой с помощью протокола HTTP. В состав клиентской части входят следующие подсистемы:

1. подсистема проектирования, состоящая из:

* подсистемы составления задания, которая позволяет учителю создавать задания для учеников, используя панель создания игрового поля и описания текстового задания;

Рисунок 10 – Структурная схема разрабатываемой системы



* подсистемы составления решения, которая позволяет ученику составить алгоритм решения из имеющихся команд, представленных графическими примитивами;
* подсистемы валидации, которая отвечает за проверку составленного задания на соответствие требованиям системы и за проверку семантики решения задания;

1. подсистема ввода данных, которая отвечает за пользовательский ввод на всех этапах функционирования системы;
2. подсистема визуализации, которая отвечает за отображение статистических данных в виде графиков, а также отображение анимированного решения задачи;
3. справочная подсистема, которая отвечает за выдачу информации о разработчиках и руководстве пользователя;
4. подсистема взаимодействия с сервером, которая предоставляет интерфейс взаимодействия клиента с сервером.

В состав серверной части входят следующие подсистемы:

1. подсистема регистрации, которая отвечает за регистрацию пользователей в системе;
2. подсистема аутентификации и авторизации, которая отвечает за аутентификацию и авторизацию каждого зарегистрированного пользователя;
3. подсистема работы со статистикой, которая отвечает за сбор статистики по каждому заданию;
4. подсистема взаимодействия с БД, которая отвечает за добавление, изменение или удаление данных о пользователях, заданиях, статистике в базе данных.
   1. Разработка спецификации требований

Разработка спецификации программного обеспечения является одним из фундаментальных процессов технологии разработки ПО. Этот процесс анализа, формирования, документирования и проверки функциональных возможностей и ограничений системы называется в терминологии программной инженерии «разработка требований» (спецификация требований) [26].

Данный процесс является критическим этапом в создании всех видов программных систем, что обусловлено тем, что ошибки, допущенные на этой стадии, ведут к возникновению серьезных проблем на этапах проектирования и разработки.

Опыт индустрии информационных технологий однозначно показывает, что вопросы, связанные с управлением требованиями, оказывают критически-важное влияние на программные проекты, в определенной степени – на сам факт возможности успешного завершения проектов. Только систематичная работа с требованиями позволяет корректным образом обеспечить моделирование задач реального мира и формулирование необходимых приемочных тестов для того, чтобы убедиться в соответствии создаваемых программных систем критериям, заданным реальными практическими потребностями [26].

Функциональные требования задают «что» система должна делать; нефункциональные – с соблюдением «каких условий» (например, скорость отклика при выполнении заданной операции). При разработке этих требований в первую очередь необходимо учитывать потребности пользователя (заказчика). Пользовательские требования (User Requirements) – описывают цели/задачи пользователей системы, которые должны достигаться/выполняться пользователями при помощи создаваемой программной системы. Часто пользовательские требования представляют в виде сценариев (вариантов использования) Use Сase [27].

Среди нефункциональных требований на первый план выходят атрибуты качества и ограничения. Атрибуты качества (Quality Attributes) описывают дополнительные характеристики продукта в различных «измерениях», важных для пользователей и/или разработчиков. Атрибуты касаются вопросов портируемости, интероперабельности (прозрачности взаимодействия с другими системами), целостности, устойчивости и т.п. Данный вид требований мы будем называть спецификацией качества. Ограничения (Constraints) включают в себя формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. В частности, к ним могут относиться параметры производительности, влияющие на выбор платформы реализации и/или развертывания, которые, в свою очередь, могут относиться, например, к внешним интерфейсам.

Спецификация требований к ПО (SRS) – процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 9126-94, которые будут отрабатываться на этапах жизненного цикла программного обеспечения [26].

* + 1. Функциональная спецификация

Функциональная спецификация включает в себя перечень всех функций системы с привязкой их к конкретной подсистеме и к информационной среде (входные и выходные данные.

Функциональная спецификация клиентской части системы приведена в таблице 2, а серверной части в таблице 3

* + 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 (клиентская часть)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название  подсистемы | Название функции | Информационная среда | | | |
| Входные данные | | Выходные данные | |
| Назначение (наименование) | Тип, ограничения | Назначение (наименование) | Тип, ограничения |
|  |  |  |  |  |  |
| 1 Справочная | 1.1 Выдать сведения о разработчиках | Сведения о разработчиках системы (ФИО, номер группы) | Текст (МЕМО) | Визуальное отображение информации | – |
| 1.2 Выдать сведения о системе | Файл справки | Текстовый (\*.HTML) |
| Код ошибки | Целое |
| 2 Ввода данных | 2.1 Ввести логин | Набор допустимых символов | Латинские буквы | Логин | Строка |
| Цифры |
| Спецсимволы | Код ошибки | Целое |
| Допустимая длина | Целое, от 4 – 16 |
| 2.2 Ввести пароль | Набор допустимых символов | Латинские буквы | Пароль | Строка |
| Русские буквы |
| Цифры | Код ошибки | Целое |
| Спецсимволы |
| Допустимая длина | Целое, от 6 до 16 |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 2 Ввода данных | 2.3 Ввести логин | Набор допустимых символов | Латинские буквы | логин | Строка |
| Цифры |
| Спецсимволы | Код ошибки | Целое |
| 2.4 Ввести ФИО | Набор допустимых символов | Русские буквы | ФИО | Строка |
| Код ошибки | Целое |
| 2.5 Выбрать роль | Список ролей | Учитель | Роль пользователя | Перечисление |
| Ученик |
| 3 Визуализации | 3.1 Отобразить игровое поле | Игровое поле | Объект «Игровое поле» | Визуальное отображение игрового поля. | – |
| 3.2 Отображение списка команд | Список команд | Вперед | Визуальное отображение списка команд | – |
| Назад |
| Поворот налево |
| Поворот направо |
| Прыжок |
| Цикл |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 3 Визуализации | 3.3 Воспроизведение алгоритма | Алгоритм | Команды | Визуальное отображение алгоритма | – |
| 3.4 Отображение статистики ученика | ID ученика | Целое | Визуальное отображение статистики ученика в виде диаграммы | – |
| Статистика | Строка |
| 4 Составления задания | 4.1 Ввести текстовое описание задания | Допустимая длина | Целое, 0 до 256 | Текстовое описание задания | Строка |
| 4.2 Выбрать режим создания игрового поля | Список режимов | Ручной | Режим создания игрового поля | Перечисление |
| Автоматический |
| 4.3 Задать сторону игрового поля | Допустимое значение | Целое, от 4 до 12 | Значение стороны поля | Целое |
| 4.4 Задать % бананов | Допустимое значение | Целое, от 0 до 30% от общего числа клеток | % бананов | Целое |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 4 Составления задания | 4.5 Задать % корзин | Допустимое значение | Целое, от 1 корзины до 10% от общего числа клеток | % норок | Целое |
| 4.6 Задать % лиан | Допустимое значение | Целое, от 0 до 20% от общего числа клеток | % лиан | Целое |
| 4.7 Выбрать тип объекта | Тип объекта | Пустая клетка | Текущий тип объекта | Объект типа «Клетка» |
| Клетка-лиана |
| Клетка-корзина |
| Клетка-банан |
| Обезьянка |
| 4.8 Определить расположение объекта | Значение стороны поля | Целое | Координаты клетки | Целое/целое |
| 4.9 Добавить объект на игровое поле | Координаты клетки | Целое/целое | Структура игрового поля | Строка |
| Текущий тип объекта | Объект типа «Клетка» |
| Структура игрового поля | Строка |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 4 Составления задания | 4.10 Очистить ИП | Структура игрового поля | Строка | Пустое игровое поле | Строка |
| 4.11 Проверить корректность ИП | Структура игрового поля | Строка | Код ошибки | Целое |
| 5 Составления решения | 5.1 Выбрать задание | Список доступных заданий | Массив строк | Текущее задание | Строка |
| 5.2 Выбрать команду | Список графических примитивов | Вперед | Текущая команда | Объект типа «Команда» |
| Назад |
| Поворот налево |
| Поворот направо |
| Прыжок |
| Цикл |
| 5.3 Добавить команду в алгоритм | Текущая команда | Объект типа «Команда» | Список команд | Список объектов типа «Команда |
| Месторасположение | Целое |
| 5.4 Задать количество итераций цикла | Допустимый диапазон | Целое, от 1 до 16 | Количество итераций | Целое |

Продолжение страницы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5.5 Выбрать команду в программе | Список команд | Список объектов типа «Команда» | Текущая команда | Объект типа «Команда» |
| 5.6 Очистить список команд | Список команд в алгоритме | Объект типа команда | Список команд | Список объектов типа «Команда» |
| 6 Валидации | 6.1 Проверить возможность установки объекта | Структура ИП | Объект типа «Игровое поле» | Признак валидности | Логическое |
| Объект | Объект типа «Клетка» |
| 6.2 Проверить возможность выполнения команды | Команда из списка | Объект «Команда | Признак валидности | Логическое |
| Структура ИП | Объект типа «Игровое поле» |
| Координаты обезьянки | Целое/целое | Координаты обезьянки | Целое/целое |
| 7 Взаимодействия с серверной частью | Отправить данные регистрации | Логин | Строка | Сведения о пользователе | Объект «Пользователь» |
| Пароль | Строка |
| ФИО | Строка |
| Роль | Перечисление |
| Получить ответ о регистрации | Код ошибки | Целое | Сообщение о статусе регистрации | Строка |

Продолжение страницы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 Взаимодействия с серверной частью | Отправить данные регистрации | Логин | Строка | Сведения о пользователе | Объект «Пользователь» |
| Пароль | Строка |
| ФИО | Строка |
| Роль | Перечисление |
| Получить ответ о регистрации | Код ошибки | Целое | Сообщение о статусе регистрации | Строка |
|  | Отправить данные для авторизации | Логин | Строка | Сведения о пользователе | Объект «Пользователь» |
| Пароль | Строка |
|  | Получить ответ об авторизации | Код ошибки | Целое | Сообщение о статусе авторизации | Строка |
|  | Назначить задание ученику | Задание | Объект типа «Задание» | – | – |
| ID ученика | Целое |
|  | Получить ответ о назначении задания ученику | Код ошибки | Целое | Сообщение о назначении задания ученику | Строка |
|  | Отправить задание на проверку учителю | Задание | Объект типа «Задание» | – | – |
| ID ученика | Целое |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 Взаимодействия с серверной частью | Получить ответ об отправке задания на проверку | Код ошибки | Объект типа «Задание» | Сообщение об отправке задания на проверку | Строка |
| Отправить запрос о получении списка учеников | – | – | ID учителя | Целое |
| Получить список учеников | – | – | Список учеников | Массив строк |
| Отправить запрос о получении статистики | – | – | ID ученика | Целое |
| Получить статистику ученика | – | – | Статистика ученика | Строка |

Таблица 4 – Перечень функций, выполняемых системой (серверная часть)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название  подсистемы | Название функции | Информационная среда | | | | |
| Входные данные | | Выходные данные | | |
| Назначение (наименование) | Тип, ограничения | Назначение (наименование) | | Тип, ограничения |
| 1 Аутентификации и авторизации | Аутентифицировать пользователей | Логин | Строка | Код ошибки | | Целое |
| Пароль | Строка | Роль | | Перечисление |
| Список учетных данных пользователей | Сущность БД «Пользователь» |
| 2 Регистрации | Зарегистрировать пользователя | Логин | Строка | Код ошибки | Целое | |
| Пароль | Строка |
| ФИО | Строка |
| Роль | Перечисление |
| Список учетных данных пользователей | Сущность БД «Пользователь» | Список учетных данных пользователей | Сущность БД «Пользователь» | |

Продолжение таблицы 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 3 Работы со статистикой | Сформировать статистику ученика | ID ученика | Целое | Статистика ученика | Строка |
| 4 Взаимодействия с клиентом | Отправить список учеников | ID учителя | Целое | Код ошибки | Целое |
| Отправить статистику ученика | Статистика ученика | Строка | Код ошибки | Целое |
| 5 Взаимодействия с БД | Сохранить информацию о пользователе | Данные учетной записи | Объект типа «Пользователь» | Код ошибки | Целое |
| Список пользователей | Сущность БД «Пользователь» |
| Получить информацию о пользователе | ID пользователя | Целое | Код ошибки | Целое |
| Список пользователей | Сущность БД «Пользователь» | Данные учетной записи | Объект типа «Пользователь» |

Продолжение таблицы 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 5 Взаимодействия с БД | Удалить пользователя | ID пользователя | Целое | Код ошибки | Целое |
| Список пользователей | Сущность БД «Пользователь» | Список пользователей | Сущность БД «Пользователь» |
| Сохранить задание | Задание | Объект типа «Задание» | Код ошибки | Целое |
| Список заданий | Сущность БД «Задание» |
| Получить иноформацию о задании | ID задания | Целое | Задание | Сущность БД «Задание» |
| Список заданий | Сущность БД «Задание» | Код ошибки | Целое |
| Сохранить статистику | Статистика ученика | Строка | Код ошибки | Целое |
| Список статистик учеников | Строка |

Продолжение таблицы 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 Взаимодействия с БД | Получить информацию о статистике ученика | ID ученика | Целое | Статистика ученика | Строка |
| Список статистик учеников | Строка | Код ошибки | Целое |

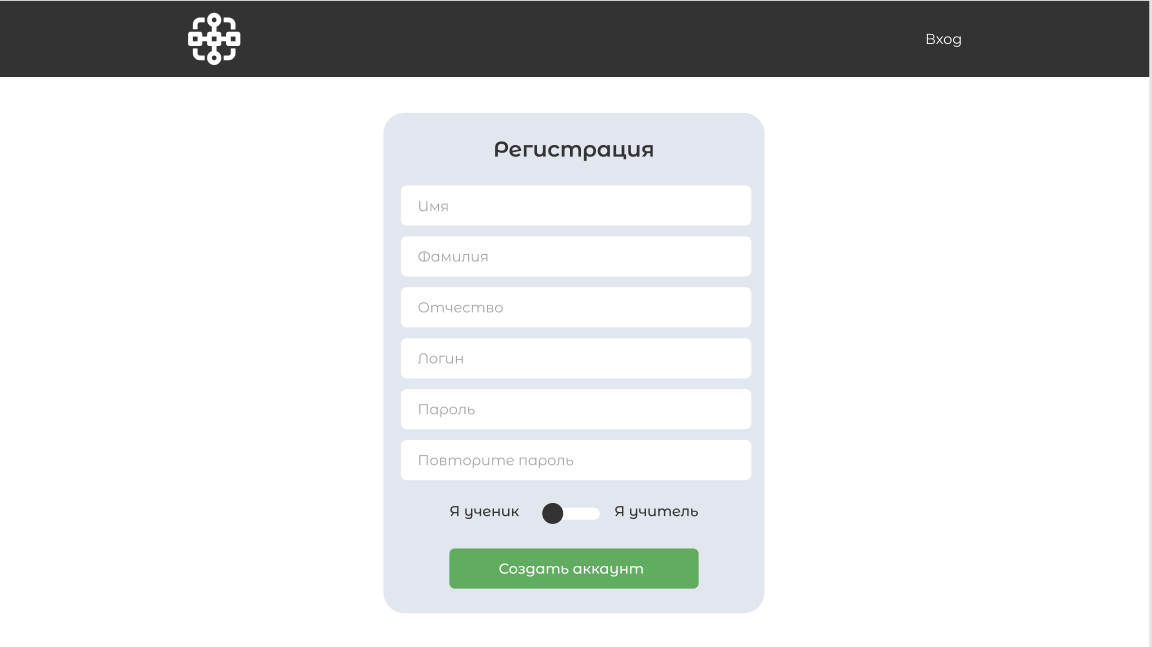
Таблица 4 – Перечень исключительных ситуаций

|  |  |  |
| --- | --- | --- |
| 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 Взаимодействия с серверной частью | 2.1 Отсутствует соединение с сервером | Выдача сообщения «Отсутствует соединение с сервером» |
| 3 Взаимодействия с базой данных | 3.1 Отсутствует соединение с базой данных | Выдача сообщения «Отсутствует соединение с базой данных» |

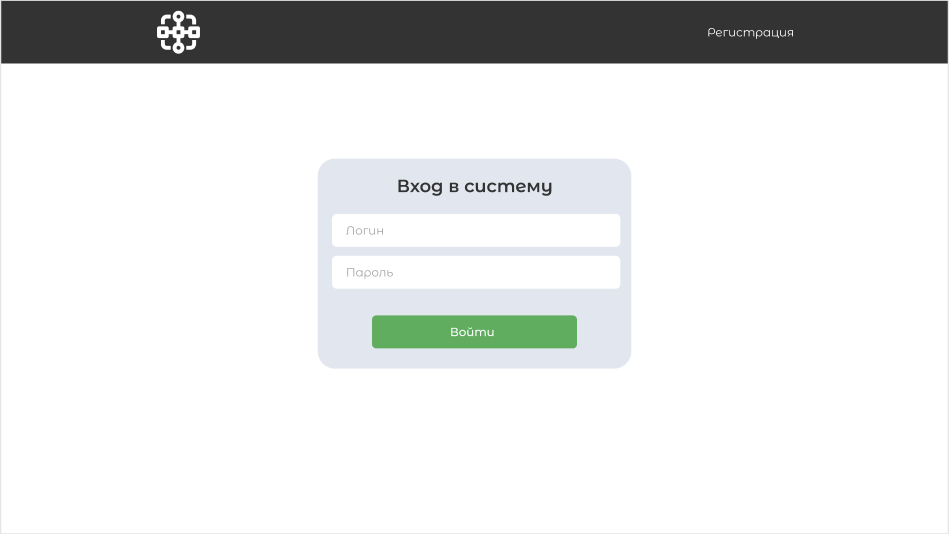
* 1. Разработка прототипа интерфейса пользователя системы

Интерфейс пользователя является важной частью системы, так как именно через работу с интерфейсом пользователь может взаимодействовать с системой. Важно проектировать интерфейс пользователя интуитивно понятным, чтобы даже при первом запуске он не вызывал никаких вопросов. Руководствуясь данным принципом, был разработан пользовательский интерфейс разрабатываемой системы.

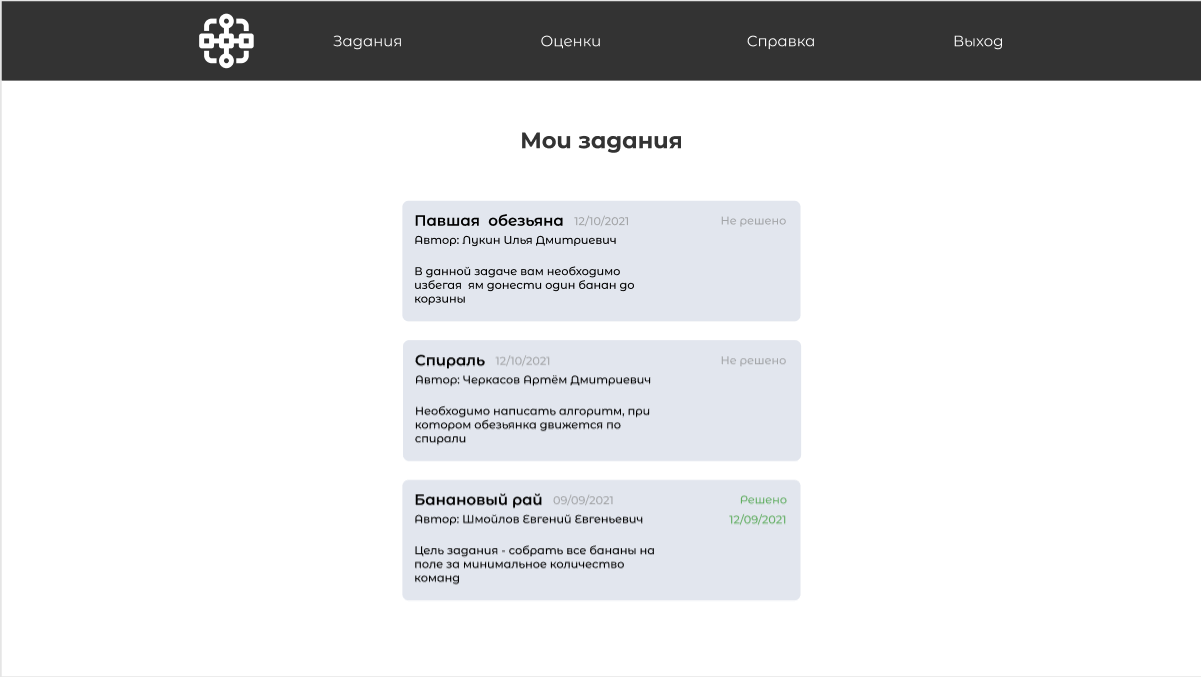
На рисунке 11 приведен прототип экранной формы регистрации. Здесь пользователь должен ввести имя, фамилию, отчество, логин, пароль, а также должен выбрать роль, под которой он хочет зарегистрироваться (учитель или ученик). Далее по нажатию кнопки «Создать аккаунт», пользователь попадает на страницу с заданиями.

  
Рисунок 11 – Прототип экранной формы регистрации

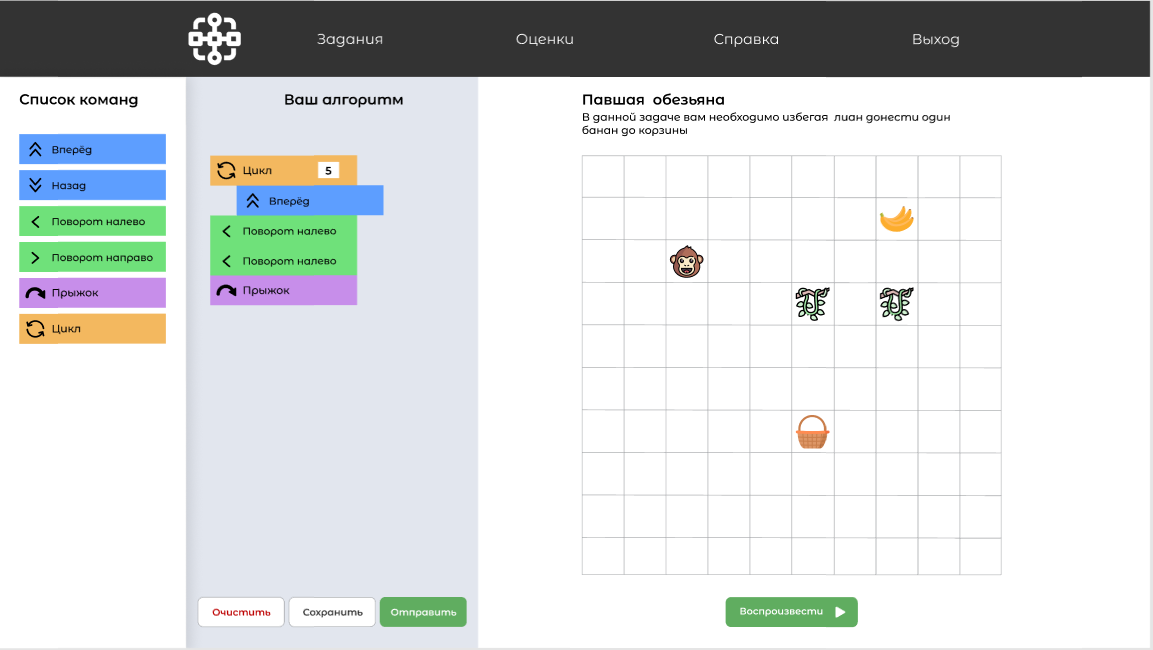
На рисунке 12 приведен прототип экранной формы входа в систему. Пользователь должен ввести логин и пароль, выбрать соответствующую роль (учитель или ученик), а затем нажать на кнопку «Войти». После этого он попадет на страницу с заданиями.

  
Рисунок 12 – Прототип экранной формы входа ученика

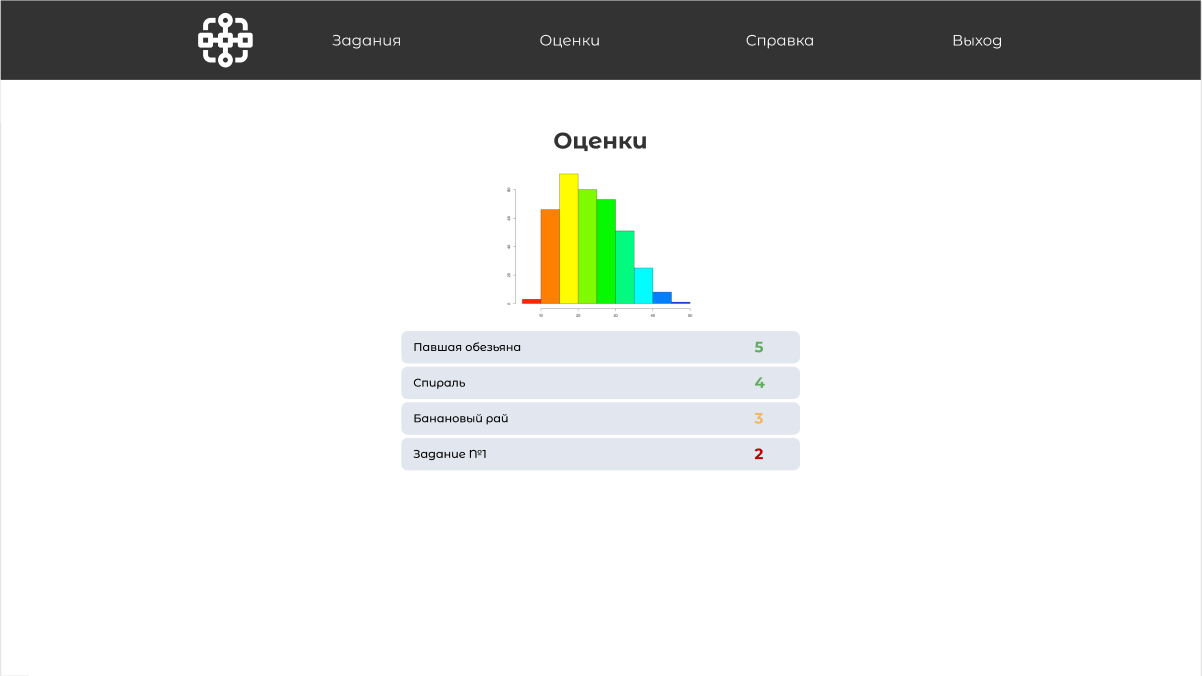
На рисунке 13 приведен прототип экранной формы заданий ученика. Ученик должен выбрать задание для выполнения из прокручиваемого списка назначенных заданий и нажать на него. После этого он перейдет на форму решения задания.

  
Рисунок 13 – Прототип экранной формы заданий ученика

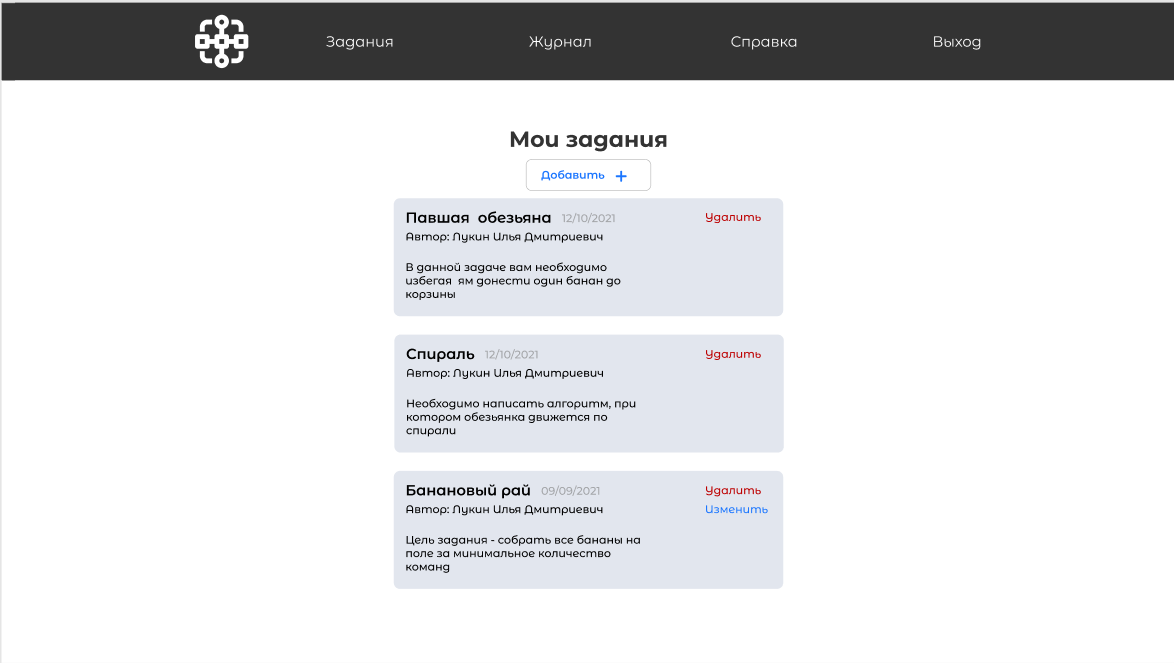
На рисунке 14 приведен прототип экранной формы решения задания ученика. Форма разделена на 3 блока: список команд, созданный алгоритм и игровое поле. Ученик должен, перетаскивая команды из списка команд, составить алгоритм решения задания. Также, в блоке с алгоритмом будут доступны кнопки «Очистить», «Сохранить» и «Опубликовать», которые соответственно очищают весь алгоритм, сохраняют текущий алгоритм и отправляют алгоритм на проверку. Ученику предоставляется возможность воспроизвести созданный алгоритм по кнопке «Воспроизвести» из блока игрового поля.

  
Рисунок 14 – Прототип экранной формы решения задачи ученика

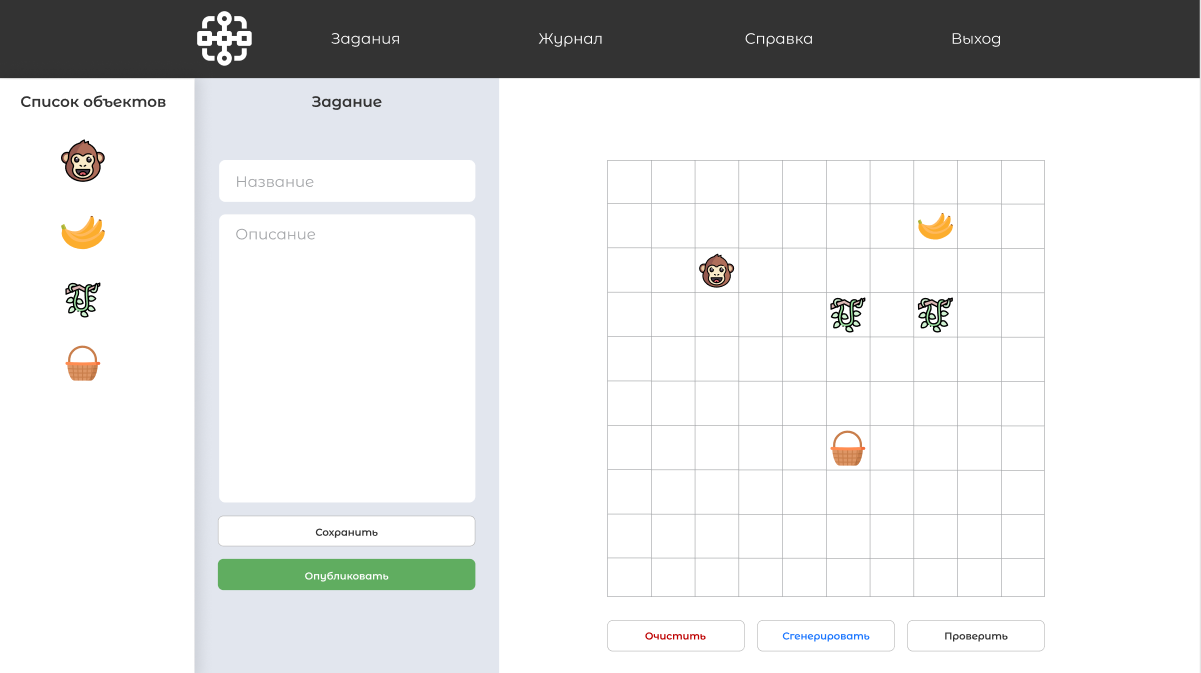
Статистику решения задач ученик сможет просмотреть, нажав на кнопку «Оценки» из навигационного меню системы. Экранная форма с оценками ученика представлена на рисунке 15. Здесь будет отображать список всех решенных задач ученика с оценками. На основе этих оценок будет строиться гистограмма, которая также отображается на этой форме.

  
Рисунок 15 – Прототип экранной формы статистики ученика

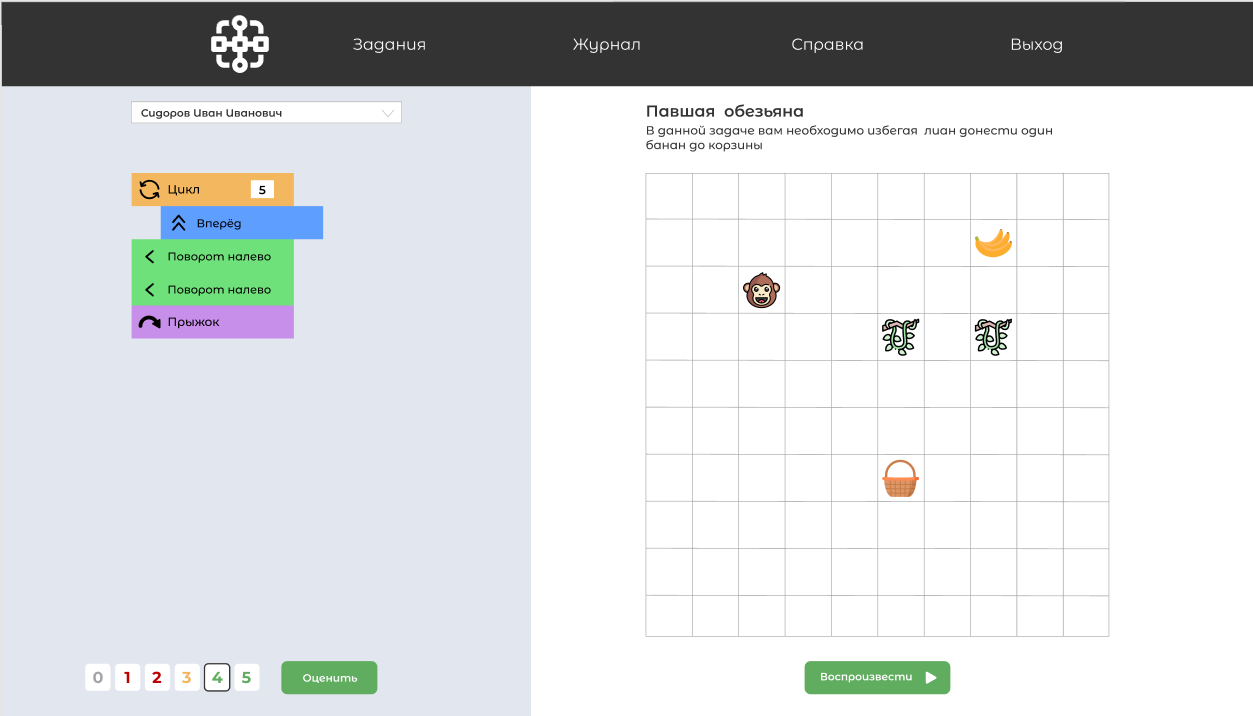
На рисунке 16 представлен прототип экранной формы заданий учителя. Здесь будет отображаться прокручиваемый список заданий учителя. Учитель должен нажать на соответствующее задание чтобы перейти на форму для проверки алгоритмов учеников к этому заданию. По нажатию кнопки «Добавить», учитель перейдет на форму с составлением задания. Также учитель может удалить любое из заданий или изменить не опубликованное задание по нажатию на соответствующие кнопки на карточке задания.

  
Рисунок 16 – Прототип экранной формы заданий учителя

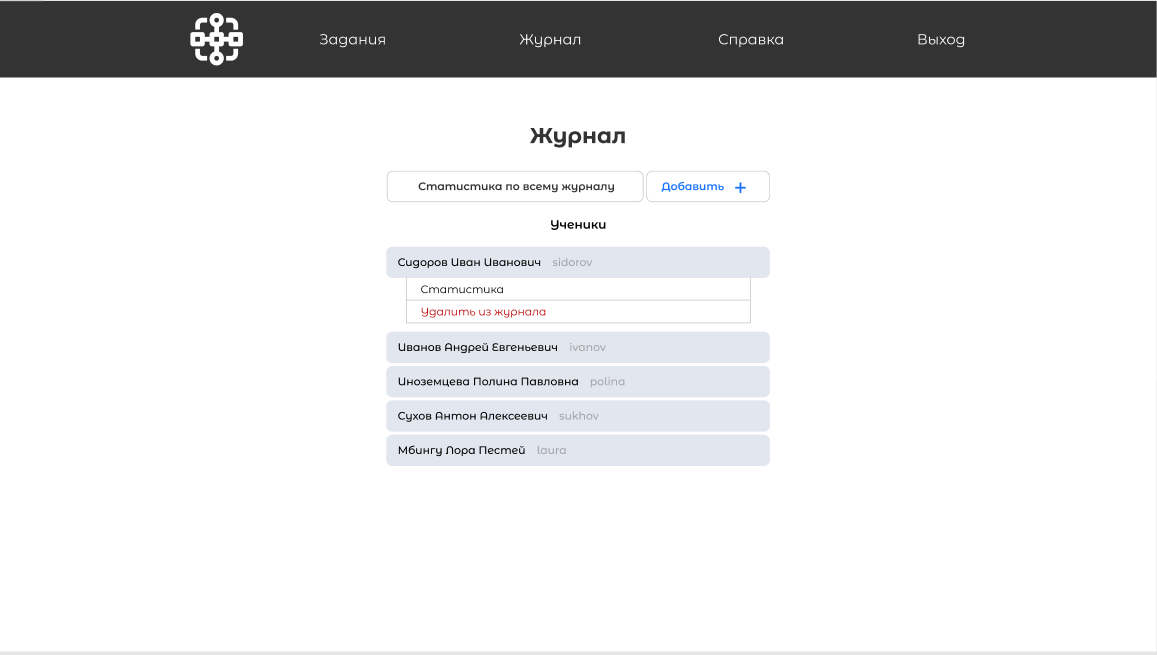
На рисунке 17 приведен прототип экранной формы конструктора заданий учителя. Форма разделена на 3 блока: блок с объектами в виде графических примитивов, блок с названием и описания задания, блок с игровым полем. Для расположения объекта на игровом поле, учитель должен перетащить графический примитив на соответствующую клетку игрового поля. Чтобы убрать объект с клетки, необходимо просто нажать на данный объект. Далее учитель должен ввести название и описание задания в соответствующие формы ввода. Учителю предоставится возможность либо сохранить задание, но не выкладывать его, либо сразу опубликовать для всех учеников. Помимо этого, возможно будет очистить всё игровое поле задания, сгенерировать поле по заданным параметрам, а также проверить данное задание, составив свой алгоритм решения.

  
Рисунок 17 – Прототип экранной формы конструктора заданий учителя

На рисунке 18 приведена экранная форма проверки задания учителем. Сначала учитель должен выбрать ФИО ученика из выпадающего списка в блоке с алгоритмом. Затем в этом же блоке появится алгоритм решения ученика, который можно воспроизвести по нажатию на кнопку «Воспроизвести». После этого, учитель должен выставить оценку от 0 до 5 и нажать на кнопку «Оценить».

  
Рисунок 18 – Прототип экранной формы проверки задания учителем

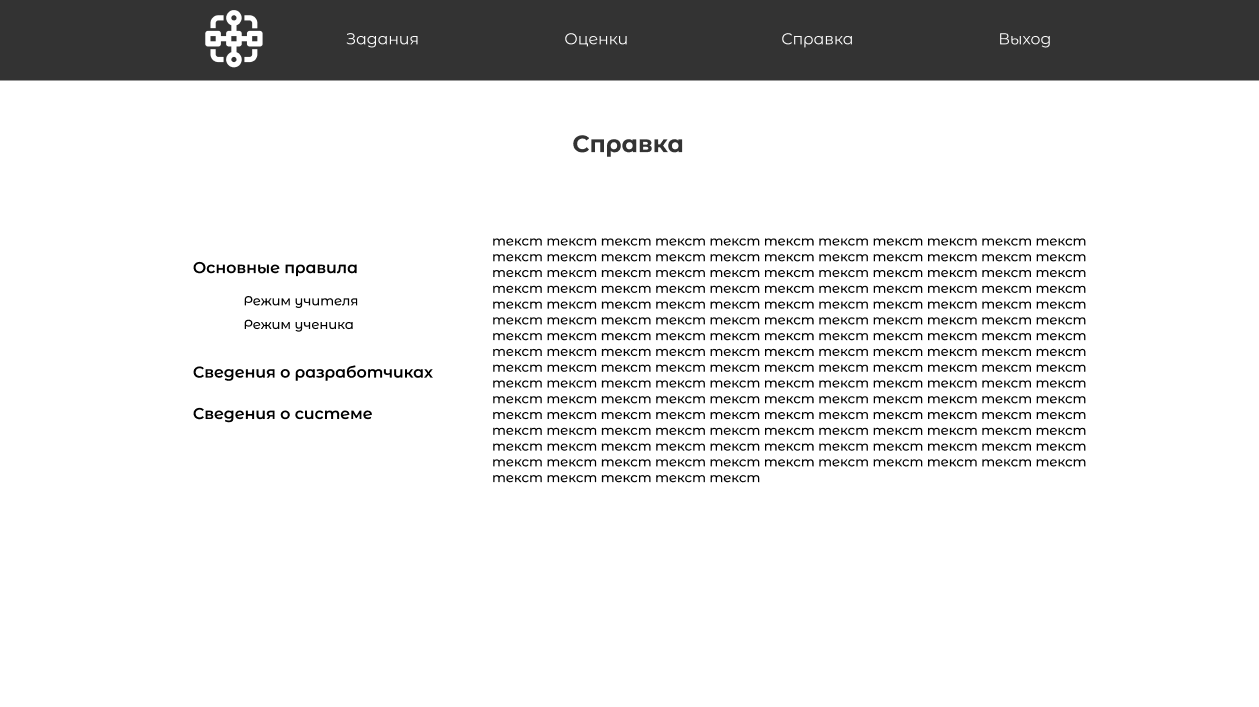
На рисунке 19 представлен прототип экранной формы журнала учителя. На данном экране учитель сможет просмотреть список учеников, а также добавить в свой журнал новых учеников по нажатию на кнопку «Добавить». Чтобы просмотреть статистику по всему журналу, необходимо будет нажать на соответствующую кнопку. Чтобы посмотреть статистику отдельного ученика или удалить его из журнала, учитель должен нажать на ученика в списке, после этого появятся кнопки «Статистика» и «Удалить из журнала».

  
Рисунок 19 – Прототип экранной формы журнала учителя

Для ознакомления со справочной информацией, панель навигации будет содержать соответствующий раздел «Справка». При нажатии на данную кнопку открывается экранная форма, прототип которой представлен на рисунке 20. На данном экране будет отображаться информация о режимах ученика и учителя, сведения о системе, а также сведения о разработчиках.

* 1. Разработка информационно-логического проекта системы

Одной из широко используемых методик документирования требований является построение ряда моделей системы.

  
Рисунок 20 – Прототип экранной формы справки

Эти модели используют графические представления, показывающие решения как исходной задачи, так и разрабатываемой системы [3]. Как правило, графическое представление более понятно, чем описание требований на естественном языке.

Моделирование – это устоявшаяся и повсеместно принятая инженерная методика. Модели являются связующим звеном между процессом анализа и процессом проектирования системы. Хорошая модель всегда включает элементы, которые существенно влияют на результат, и не включает те, которые малозначимы на данном уровне абстракции [3].

Моделирование предназначено не только для создания больших систем. От моделирования может выиграть любой проект. Даже при создании одноразовых программ, когда зачастую бывает полезно выбрать неподходящий код из-за преимущества в скорости разработки, которое дают языки визуального программирования, моделирование поможет коллективу разработчиков лучше представить план системы, а значит, выполнить проект быстрее и создать именно то, что подразумевал первоначальный замысел. Чем сложнее проект, тем более вероятно, что из-за отсутствия моделирования он свернется раньше времени – или будет создано не то, что нужно. Все полезные и интересные системы с течением времени обычно усложняются. Пренебрегая моделированием в самом начале создания системы можно серьезно пожалеть об этом, когда будет уже слишком поздно [3].

* + 1. Язык UML

Для специфицирования (построения точных, недвусмысленных и полных моделей) системы и ее документирования используется унифицированный язык моделирования UML.

UML можно использовать для визуализации, спецификации, конструирования и документирования артефактов программных систем. UML подходит для моделирования любых систем – от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени [3].

Язык UML не является языком программирования (он инвариантен к языкам программирования). В нотации языка UML определены следующие виды канонических диаграмм[3]:

* вариантов использования (usecasediagram);
* классов (class diagram);
* кооперации (collaboration diagram);
* последовательности (sequence diagram);
* состояний (statechart diagram);
* деятельности (activity diagram);
* компонентов (component diagram);
* развертывания (deployment diagram).
  + 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.

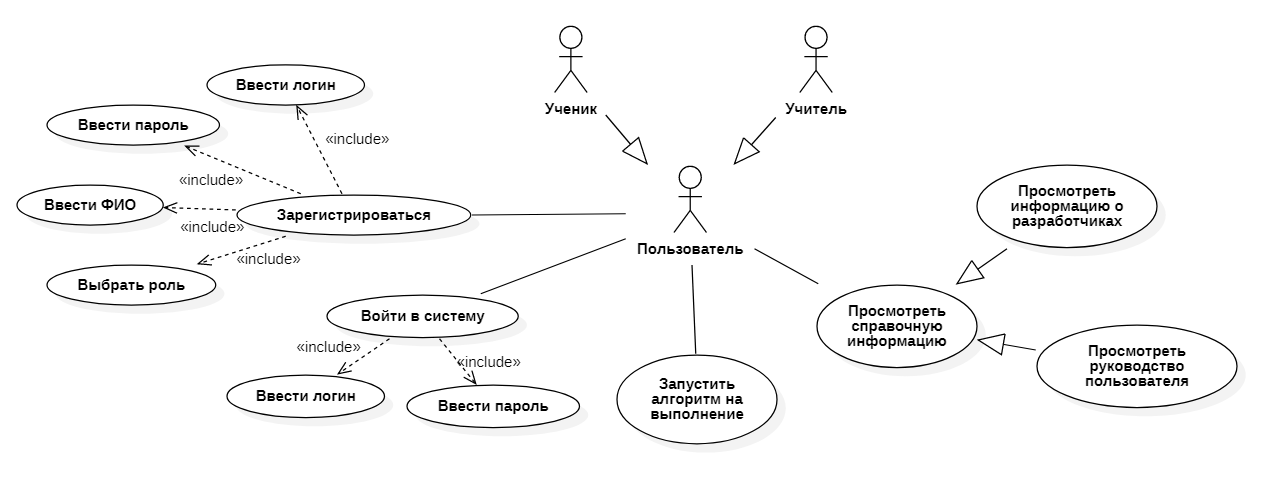
Актер (actor) – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними (это может быть любой объект, субъект или система, взаимодействующая с моделируемой бизнес-системой извне, т.е. человек, техническое устройство, программа и т.п.).

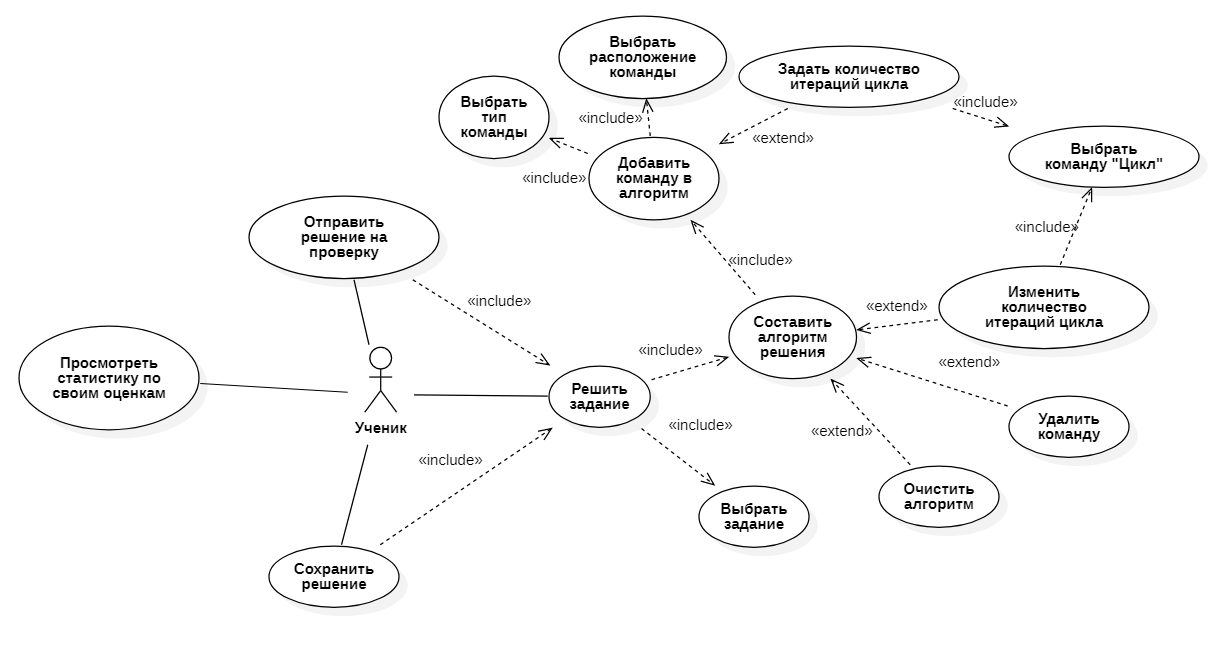
Вариант использования – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами (он определяет набор действий, совершаемый системой при диалоге с актером). Цель спецификации варианта использования заключается в том, чтобы зафиксировать некоторый аспект или фрагмент поведения проектируемой системы без указания особенностей реализации данной функциональности [2].

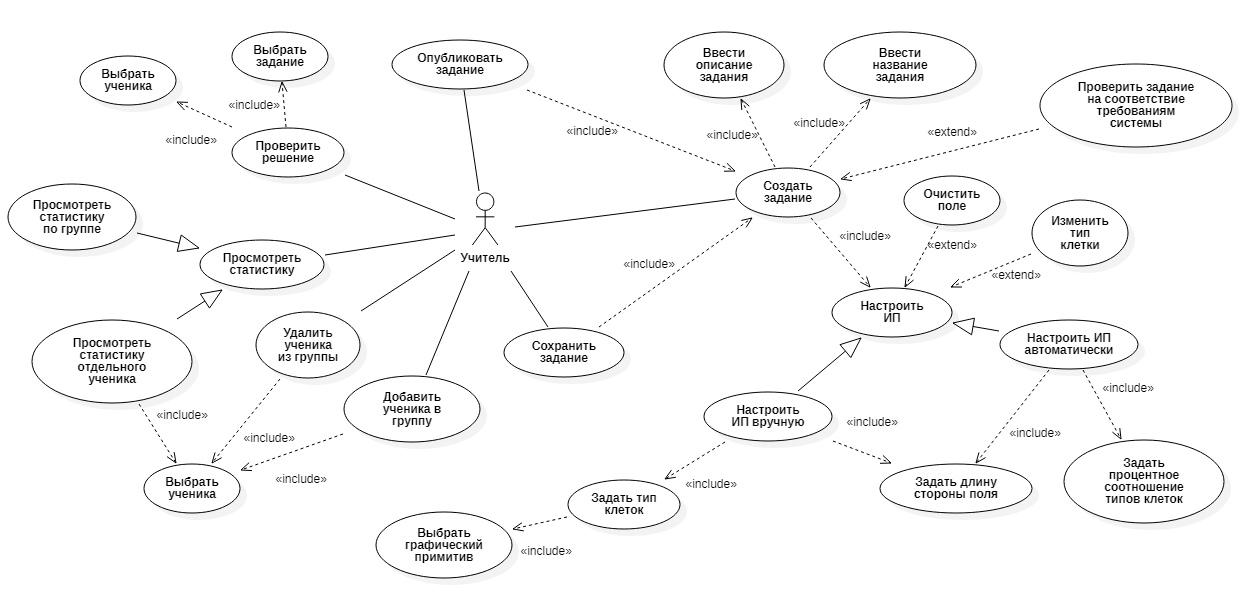
На рисунке 21 представлена диаграмма вариантов использования системы для пользователя. К общим для ученика и учителя вариантам использования относятся: регистрация, вход в систему, отображение файла справки и визуальное отображение решения задания.

На рисунке 22 представлена диаграмма вариантов использования системы для ученика. После успешной авторизации ученику доступны функции, определяющиеся его ролью: решить задание, сохранить задание, отправить задание на проверку, а также просмотреть статистику по своим оценка.

На рисунке 23 представлена диаграмма вариантов использования системы для учителя. После успешной авторизации учителю доступны функции, определяющиеся его ролью: добавить ученика в группу, удалить ученика из группы, создать задание, сохранить задание, опубликовать задание, проверить решение учеников, а также просмотреть статистику по всему журналу, либо по отдельному ученику.

  
Рисунок 21 ‒ Диаграмма вариантов использования (пользователь)

  
Рисунок 22 ‒ Диаграмма вариантов использования (ученик)

  
Рисунок 23 ‒ Диаграмма вариантов использования (учитель)

* + 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.

Сценарии определяются преподавателем.

* + 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 [ХХХ].

На рисунке ХХ приведена диаграмма классов системы (этап проектирования). В таблице ХХ приведено описание классов.

Таблица ХХ – Описание классов системы

|  |  |
| --- | --- |
| Название класса | Назначение |
| 1 | 2 |
|  |  |

Продолжение таблицы ХХ

|  |  |
| --- | --- |
| 1 | 2 |
|  |  |
|  |  |
|  |  |

* + 1. Диаграмма состояний

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

На рисунке ХХХ приведена диаграмма состояний системы. Здесь должно быть описание диаграммы (диаграмм).

* + 1. Диаграмма деятельности

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

На рисунке ХХХ приведена диаграмма деятельности системы. Здесь должно быть описание диаграммы (диаграмм).

* + 1. Диаграмма последовательности

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

На рисунке ХХХ приведены диаграммы последовательности системы для вариантов использования «???». Диаграммы построены на основании сценариев, приведенных в п.2.4.3.

* 1. Логическая модель данных (при необходимости)

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 [**Ошибка! Источник ссылки не найден.**].

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 [**Ошибка! Источник ссылки не найден.Ошибка! Источник ссылки не найден.**].

Логическая модель БД разрабатываемой системы приведена на рисунке ХХХ.

  
Рисунок 1 – Логическая модель данных

Описание объектов рассматриваемой предметной области, которые хранятся в базе данных, приведено в таблицах 2-???.

Таблица 2 – 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[10] | Пароль пользователя, преобразованный в закодированную строку |
| Email | Символьный[50] | Электронная почта, указанная пользователем при регистрации |

* 1. Выбор и обоснование алгоритмов обработки данных /Разработка и описание алгоритмов обработки данных

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* 1. Выбор и обоснование комплекса программных средств

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Выбор языка программирования

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст. Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Выбор операционной системы

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Выбор среды программирования

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Выбор системы управления базами данных (при необходимости)

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

1. Реализация системы
   1. Разработка и описание интерфейса пользователя
      1. Разработка и описание пользовательского меню

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Описание тестового примера

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* 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) [ХХХ].

* + 1. Диаграмма компонентов

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Диаграмма развертывания

Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст Текст.

* + 1. Диаграмма классов

В соответствии со спецификацией, приведенной в п. 2.5.6, и с учетом выбранного языка программирования (см. п. 2.8.1) разработана диаграмма классов системы (этап реализации), приведенная на рисунке ХХХ.

Рисунок ХХХ – Описание классов системы (этап реализации)

* 1. Физическая модель данных (при необходимости)

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).

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

На рисунке **Ошибка! Источник ссылки не найден.** представлена физическая модель данных системы.

Рисунок ХХХ – Физическая модель данных системы

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

Таблица ХХХ – Сущность « User »

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Имя атрибута | Тип | Размер (байт) |
| **user Id** | **uniqueidentifier** | **int** | **4** |
| Name | Имя пользователя | varchar(30) | 30 |
| Password | Пароль | varchar(10) | 10 |
| e-mail | Адрес электронной почты | varchar(50) | 50 |
| Размер записи | | | 94 |

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

Расчет объема внешней памяти

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

VЖД = VОС + VПР + VСПО + VБД + Vсправки,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 7 Professional 64 бит с пакетом обновлений SP1,   
VОС = 20 Гб);

VПР – объем памяти, занимаемый непосредственно файлами приложения (VПР = 2 Мб);

VСПО – объем памяти, занимаемый сопутствующим программным обеспечением (библиотеки cryptopp.dll, simplexlsx.dll, sqlite3.dll, sqlitecpp.dll, Qt Framework 5.11.1, Internet Explorer 9; дадим оценку сверху VСПО в 3 Гб);

VБД – объем памяти, занимаемый базой данных (всеми таблицами) при ее максимальном заполнении. Расчет этой составляющей приведен в таблице ХХХ (VБД = ???? байт = ??? Кб = ??? Мб = ??? Гб).

Vсправки – объем памяти, необходимый для хранения файла справки, необходимого для работы программы (дадим ему оценку сверху в 1 Мб).

Таким образом, суммарный объем внешней памяти составит:

VЖД = 20 Гб + 2 Мб + 3 Гб + ??? Мб + 1 Мб ~ ??? Гб.

Таблица 1 – Расчет объема внешней памяти, необходимой для хранения БД

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Размер записи (байт) | Максимум записей | Всего (байт) |
| Пользователь | 94 | 10 | 940 |
| Сотрудник |  | 30 |  |
| Статус сотрудника |  | 10 |  |
| Должность сотрудника |  | 10 |  |
| Место работы |  | 10 |  |
| Кафедра |  | 10 |  |
| ОУ ВО |  | 10 |  |
| Итого | | |  |

Расчет объема ОЗУ

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + VПР + VБД + Vбраузера,

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

VПР – ОЗУ, которое займет само приложение (не превысит 80 Мб);

VБД – объем данных из базы, который может быть одновременно загружен в оперативную память (дадим ему оценку сверху в 10 Мб).

Vбраузера – ОЗУ, занимаемое браузером (оценим его сверху значением в 100 Мб).

Суммарные объемы ОЗУ составит:

VОЗУ = 2 Гб + 80 Мб + 10 МБ + 100 Мб ~ 2.2 Гб.

Таким образом, 2.2 Гб оперативной памяти можно счесть минимально необходимым для функционирования системы.

* + 1. Минимальные требования, предъявляемые к системе

Для корректного функционирования системы необходимо:

* тип ЭВМ: x86-64 совместимый;
* объем ОЗУ – не менее 3 Гб;
* объем свободного дискового пространства – не менее ??? 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;
* браузер Internet Explorer 9 и выше;
* Qt framework 5.11 и выше.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта была разработана автоматизированная система …, позволяющая ….

В первом разделе приведены основные понятия предметной области, описаны характеристики систем-аналогов, приведен их сравнительный анализ. На основе проведенного анализа выполнена объектная декомпозиция, отраженная в диаграмме объектов, и сформулирована постановка задачи.

Во втором разделе …

В третьем разделе …

Разработанная система может использоваться …

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**Книги**

***Целиком***

Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Изд. 2-е. М.: ДМК Пресс, 2006. 546 с.

*Если нужно указать номера конкретных страниц*

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

*Если повторная ссылка на тот же документ*

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

*Если больше 3 авторов*

Нестационарная аэродинамика баллистического полета/ Липницкий Ю.М. и [др.]. М.: Физматлит, 2003. 176 с.

**Журналы**

Зеленко Л.С., Шумская Е.А. Комплекс программ для работы с учебным контентом в дистанционных обучающих системах// Известия СНЦ РАН. 2015. №2 (5). Т. 17. С. 992-1003.

**Руководящие материалы и ГОСТы**

РД 34.20.571. Методические указания по расчету показателей готовности к работе электростанции и энергосистем. Введ. 1976-10-22. М., 1976. 25 с.

ГОСТ Р 7.0.4-2006. Издания. Выходные сведения. Общие требования и правила оформления. М., 2006. II. 43 с. (Система стандартов по информ., библ. и изд. делу).

**Методические указания или учебные пособия**

Зеленко Л.С. Методические указания к лабораторному практикуму по дисциплине «Программная инженерия». Самара: СГАУ, 2012. 67 с.

**Электронные ресурсы**

Российская гидроэнергетика [Электронный ресурс] // Русгидро: [сайт]. URL: http://www.rushydro.ru/industry/russianhydropower/ (дата обращения: 20.12.2021).

Гидроэлектростанция (гидроэлектрическая станция, ГЭС) // Энциклопедический словарь юного техника М.: Издательство «Педагогика», 1987 [Электронный ресурс] // Библиотекарь.Ру: электрон. библ. 2006-2021. URL: http://www.bibliotekar.ru/enc-Tehnika/58.htm (дата обращения: 20.12.2021).

Субботин А.С. Основы гидротехники [Электронный ресурс]. URL: http://www.cawater-info.net/bk/dam-safety/files/subbotin.pdf (дата обращения: 03.02.2021).

Филиальная структура компании [Электронный ресурс] // Системный оператор Единой энергетической системы: [сайт]. [2009-2017]. URL: http://so-ups.ru/index.php?id=about (дата обращения: 20.12.2021).

Автоматизированные системы управления технологическими процессами гидроэлектростанции [Электронный ресурс] // Микроника. Инжиниринговый центр: [сайт]. [1999-2016]. URL: http://mikronika-energo.ru/products/asutp/ges-asu-tp/ (дата обращения: 24.12.2021).

Автоматизированная система управления производственными процессами [Электронный ресурс] // MEScontrol: [сайт]. [2003-2017]. URL: http://mescontrol.ru/articles/systems (дата обращения: 02.04.2021).

Пушников А.Ю. Введение в системы управления базами данных: учеб. пособие [Электронный ресурс] // CITForum: электрон. библиотека. 1997-2017. URL: https://citforum.ru/database/dblearn/ dblearn06.shtml (дата обращения: 20.12.2021).

Пользовательский интерфейс [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: https://ru.wikipedia.org/ wiki/Пользовательский\_интерфейс (дата обращения: 17.03.2021).

***Если необходимо указать системные требования для доступа к документу (наличие специального ПО), то***

Белова С.В. Язык UML. Диаграмма вариантов использования. Систем. требования: PowerPoint. URL: nkse.ru/component/k2/item/  
download/7\_754f5a247edc6ec6be78218f187338a5.html (дата обращения: 17.10.2021).

**Сборники научных трудов или трудов конференций**

Философия культуры и философия науки: проблемы и гипотезы: межвуз. сб. науч. тр./ Саратов. гос. ун-т; [под ред. С.Ф. Мартыновича]. Саратов: изд-во Сарат. ун-та, 1999. 199 с.

Акимова А.Е., Трешников А.А., Зеленко Л.С. Информационная среда ГЭС. Подсистема расчета показателей эффективности работы оборудования // Перспективные информационные технологии (ПИТ-2017): сб. науч. тр. межд. научно-техн. конф.; [под ред. С.А. Прохорова]. Самара: Изд-во СНЦ РАН, 2017. С. 41-44.

***Если электронное издание***

Акимова А.Е., Трешников А.А., Зеленко Л.С. Подсистема расчета показателей эффективности работы оборудования // Математика. Компьютер. Образование: труды XXIV межд. конф., 23-28 января 2017 г., г. Пущино. URL: http://www.mce.su/rus/presentations/ p283063/ (дата обращения: 02.03.2017).

ПРИЛОЖЕНИЕ А  
Руководство пользователя

А.1 Назначение системы

Приводится краткое описание возможностей системы.

А.2 Условия работы системы

***Пример.***

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

1. Требования к техническому обеспечению:

* ЭВМ типа IBM PC;
* процессор типа x86 или x64 тактовой частоты 1400 МГц и выше;
* …

1. Требования к программному обеспечению:

* Windows 7 Professional 64 бит с пакетом обновлений SP1 и выше;
* установленная платформа .Net версии 4.0 и выше;
* установленная СУБД ….

А.3 Установка системы

***Пример.***

Система поставляется в виде zip-архива. Данный файл необходимо распаковать в любую директорию на жестком диске. Запускаемым файлом системы является файл ххх.exe.[[2]](#footnote-2)

А.4 Работа с системой

А.4.1 Работа с системой в режиме администратора (если необходимо)

Вход в систему (авторизация)

…

А.4.2 Работа с системой в режиме пользователя

Вход в систему (авторизация)

Вход в систему (регистрация)

Настройка параметров кроссворда

ПРИЛОЖЕНИЕ Б   
Листинг модулей программы

7-10 страниц исходного кода шрифт Times New Roman 10 пт 1 интервал

1. *Количество страниц, рисунков, таблиц указывается с учетом приложений* [↑](#footnote-ref-1)
2. Если необходимы дополнительные ресурсы для обеспечения работоспособности системы, то все для них также должны быть перечислены условия установки. *Если установка нестандартная, то она должна быть подробно описана (в объеме, достаточном для понимания пользователя).* [↑](#footnote-ref-2)