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 «Автоматизированная система генерирования

структуры лабиринта и нахождения выхода из него»

Обучающийся Е.А. Балашова

Обучающийся В.А. Гриднева

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 № 6402-020302D

Е.А. Балашовой

В.А. Гридневой

Тема проекта:«Автоматизированная система генерирования структуры

лабиринта и нахождения выхода из него»

1. Исходные данные к проекту**:** см. приложение к заданию
2. Перечень вопросов, подлежащих разработке:
   1. Произвести анализ предметной области: изучить основные принципы составления лабиринтов, изучить алгоритмы генерации лабиринтов и их решения
   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
3. Перечень графических разработок:
   1. Структурная схема системы
   2. Канонические диаграммы UML
   3. Схемы основных алгоритмов
4. 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Балашова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Гриднева

ПРИЛОЖЕНИЕ  
к заданию на курсовую работу  
обучающимся в группе № 6402-020302D  
Е.А. Балашовой

В.А. Гридневой

Тема проекта: «Автоматизированная система генерирования структуры лабиринта и нахождения выхода из него»

Исходные данные к проекту:

1. Характеристика объекта автоматизации:

## объект автоматизации: лабиринт;

## виды автоматизируемой деятельности:

* + процесс авторизации и регистрации пользователей;
  + процесс генерирования лабиринта в соответствии с алгоритмами;
  + процесс расстановки входа и выхода;
  + процесс автоматического прохождения лабиринта;
  + процесс визуализации нахождения выхода из лабиринта;

## количество ролей пользователей – 2;

## минимальный размер лабиринта по вертикали – 7;

## максимальный размер лабиринта по вертикали – 21;

## минимальный размер лабиринта по горизонтали – 7;

## максимальный размер лабиринта по горизонтали – 21;

## количество входов – 1;

## количество выходов – 1;

## количество способов расстановки входа и выхода – 2;

## количество алгоритмов создания лабиринта – 2;

## количество тем оформления лабиринта – 4;

## количество алгоритмов прохождения лабиринта – 2;

## количество скоростных режимов перемещения персонажа – 4;

## минимальная длина логина – 6 символов;

## максимальная длина логина – 16 символов;

## минимальная длина пароля – 6 символов;

## максимальная длина пароля – 20 символов;

1. Требования к информационному обеспечению:
2. информационное обеспечение разрабатывается на основе следующих источников:
   * лабиринты: классификация, генерирование, поиск решений [Электронный ресурс]. URL: https://habr.com/ru/articles/445378/ (дата обращения: 16.09.2024);
   * алгоритмы генерации лабиринтов и нахождения пути [Электронный ресурс]. URL: https://tproger.ru/articles/maze-generators (дата обращения: 16.09.2024);
   * генерация лабиринтов: алгоритм Эллера [Электронный ресурс]. URL: https://habr.com/ru/articles/667576/ (дата обращения: 16.09.2024);
3. требования к построению лабиринта:
   * лабиринт имеет формат 2D;
   * по периметру лабиринта – стена;
   * толщина стен равна толщине прохода;
   * вход и выход находится на периметре лабиринта, но не в углах;
   * вход и выход не совпадают;
   * стены не должны образовывать изолированные части;
   * наличие тупиков;
   * наличие пути;
4. лабиринты хранятся в файлах, структура файла определяется в процессе проектирования.
5. Требования к техническому обеспечению:
   1. Требования к техническому обеспечению серверной части:
6. тип ЭВМ – IBM PC совместимый;
7. объем ОЗУ – не менее 2 Гб;
8. объем свободного пространства на внешнем диске – не менее 50 Гб;
9. наличие подключения к сети Интернет;
10. манипулятор – мышь;
11. технические характеристики определяются в процессе выполнения проекта;
    1. Требования к техническому обеспечению клиентской части:
12. тип ЭВМ – IBM PC совместимый;
13. монитор с разрешающей способностью не ниже 800 х 600;
14. манипулятор – мышь;
15. технические характеристики определяются в процессе выполнения проекта.
16. Требования к программному обеспечению:
    1. Требования к программному обеспечению серверной части:
17. тип операционной системы – Windows 10 и выше.
    1. Требования к программному обеспечению клиентской части:
18. тип операционной системы – Windows 10 и выше;
19. браузер – Google Chrome 86.0.4240.183 (64-битный) и выше.
    1. Требования к программному обеспечению рабочего места разработчика:
20. тип операционной системы – Windows 10 и выше;
21. язык программирования – С#;
22. среда программирования – Visual Studio 2022;
23. среда проектирования – StarUML 5.2.0.
24. Общие требования к проектируемой системе:

5.1 Функции, реализуемые системой:

1. функции системы:
   * аутентификация пользователя в системе, настройка интерфейса пользователя на заданную роль;
   * генерирование шаблона лабиринта по заданному размеру;
   * случайная расстановка входа и выхода;
   * контроль расстановки входа и выхода;
   * создание структуры лабиринта по заданному алгоритму;
   * проверка структуры лабиринта при загрузке;
   * визуализация генерации лабиринта;
   * визуализация прохождения лабиринта (видимая дорожка);
   * выдача справочной информации;
2. функции администратора:
   * авторизация пользователя в системе (ввод логина и пароля);
   * настройка параметров лабиринта при создании:
3. задание размера по горизонтали;
4. задание размера по вертикали;
5. выбор способа расстановки входа и выхода;
6. выбор алгоритма генерации лабиринта;
7. настройки темы оформления;
   * ручная расстановка входа и выхода;
   * сохранение лабиринта в файл заданной структуры;
   * просмотр лабиринта;
   * просмотр справочной информации;
8. функции игрока:
   * регистрация пользователя в системе (ввод логина и пароля);
   * авторизация пользователя в системе (ввод логина и пароля);
   * выбор лабиринта;
   * загрузка лабиринта из файла;
   * изменение темы оформления лабиринта;
   * выбор алгоритма прохождения лабиринта;
   * ручное прохождение лабиринта[[1]](#footnote-1);
   * выбор алгоритма прохождения лабиринта;
   * выбор режима прохождения лабиринта;
   * выбор задержки перемещения персонажа (для автоматического режима);
   * запуск алгоритма
   * просмотр справочной информации.

5.2 Технические требования к системе:

1. режим работы – диалоговый;
2. время автоматической генерации лабиринта – не более 5 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 13.09.2024 В.А. Гриднева

13.09.2024 Е.А. Балашова

РЕФЕРАТ

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

Графическая часть: ??? слайдов презентации PowerPoint.

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

Объектом автоматизации является лабиринт.

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.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 12](#_Toc178706658)

[1 Описание и анализ предметной области 14](#_Toc178706659)

[1.1 Описание предметной области 14](#_Toc178706660)

[1.1.1 Описание лабиринта 14](#_Toc178706661)

[1.1.2 Классификация лабиринтов 14](#_Toc178706662)

[1.1.3 Описание алгоритмов генерации лабиринта 22](#_Toc178706668)

[1.1.4 Описание алгоритмов нахождения пути 24](#_Toc178706676)

[1.2 Описание систем-аналогов 25](#_Toc178706680)

[1.2.1 Gotcha 26](#_Toc178706684)

[1.2.2 Онлайн генератор лабиринтов «Plottersvg» 27](#_Toc178706688)

[1.3 Диаграмма объектов предметной области 28](#_Toc178706696)

[1.4 Постановка задачи 31](#_Toc178706697)

[1.4.1 Режим администратора 32](#_Toc178706698)

[1.4.2 Режим игрока 32](#_Toc178706699)

[2 Проектирование системы 35](#_Toc178706700)

[2.1 Выбор и обоснование архитектуры системы 35](#_Toc178706701)

[2.2 Структурная схема системы 37](#_Toc178706702)

[2.3 Разработка спецификации требований 39](#_Toc178706703)

[2.3.1 Функциональная спецификация 40](#_Toc178706704)

[2.3.2 Перечень исключительных ситуаций 41](#_Toc178706705)

[2.4 Разработка прототипа интерфейса пользователя системы 46](#_Toc178706706)

[2.5 Разработка информационно-логического проекта системы 50](#_Toc178706707)

[2.5.3 Язык UML 52](#_Toc178706708)

[2.5.4 Диаграмма вариантов использования 52](#_Toc178706709)

[2.5.5 Сценарии 52](#_Toc178706710)

[2.5.6 Диаграмма классов 54](#_Toc178706711)

[2.5.7 Диаграмма состояний 54](#_Toc178706712)

[2.5.8 Диаграмма деятельности 54](#_Toc178706713)

[2.5.9 Диаграмма последовательности 55](#_Toc178706714)

[2.6 Логическая модель данных (при необходимости) 56](#_Toc178706715)

[2.7 Выбор и обоснование алгоритмов обработки данных /Разработка и описание алгоритмов обработки данных 57](#_Toc178706716)

[2.8 Выбор и обоснование комплекса программных средств 59](#_Toc178706717)

[2.8.1 Выбор языка программирования 59](#_Toc178706718)

[2.8.2 Выбор среды программирования 60](#_Toc178706719)

[2.8.3 Выбор операционной системы 60](#_Toc178706720)

[2.8.4 Выбор системы управления базами данных (при необходимости) 60](#_Toc178706721)

[3 Реализация системы 61](#_Toc178706722)

[3.1 Разработка и описание интерфейса пользователя 61](#_Toc178706723)

[3.2 Диаграммы реализации 62](#_Toc178706724)

[3.2.1 Диаграмма компонентов 62](#_Toc178706725)

[3.2.2 Диаграмма развертывания 63](#_Toc178706726)

[3.2.3 Диаграмма классов 63](#_Toc178706727)

[3.3 Физическая модель данных (при необходимости) 64](#_Toc178706728)

[3.4 Выбор и обоснование комплекса технических средств 65](#_Toc178706729)

[3.4.4 Расчет объема занимаемой памяти 65](#_Toc178706730)

[3.4.5 Минимальные требования, предъявляемые к системе 67](#_Toc178706731)

[ЗАКЛЮЧЕНИЕ 68](#_Toc178706732)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 69](#_Toc178706733)

[ПРИЛОЖЕНИЕ А Руководство пользователя 74](#_Toc178706734)

[А.1 Назначение системы 74](#_Toc178706735)

[А.2 Условия работы системы 74](#_Toc178706736)

[А.3 Установка системы 74](#_Toc178706737)

[А.4 Работа с системой 75](#_Toc178706738)

[А.4.1 Работа с системой в режиме администратора (если необходимо) 75](#_Toc178706739)

[А.4.2 Работа с системой в режиме пользователя 75](#_Toc178706740)

[ПРИЛОЖЕНИЕ Б Листинг модулей программы 76](#_Toc178706741)

ВВЕДЕНИЕ

Слово «лабиринт» пришло в русский язык из немецкого и является его прямым переводом. Первоисточник этого слова – греческий язык, в котором под лабиринтом понималось обширное пространство, состоящее из многочисленных комнат, коридоров и переходов, расположенных по запутанному плану, составленному с целью запутать незнающего человека. Другим значением этого слова считалось подземелье – природные подземные ходы, в которых можно было легко запутаться и потеряться [1].

Лабиринты являются частью культуры разных стран и эпох. Например, при разговоре об античной эпохе слово «лабиринт» ассоциируется с островом Крит и Минотавром. В Древнем Египте лабиринты строились в центре городов и использовались фараонами для управления страной и в религиозных целях. В Индии и Китае считалось, что лабиринты защищают от злых духов, поэтому входы в дома и города обязательно строили в виде лабиринтов.

В современном мире лабиринты имеют более развлекательных характер. Их используют в декоративных целях в ландшафтном дизайне, для тренировки мозга в компьютерных и настольных играх. Исследования показывают, что прохождение лабиринтов учит детей ориентироваться в пространстве и развивает память [2].

Во многих компьютерных играх встречаются лабиринты. Они могут быть как отдельным уровнем в логических играх, так и частью мира в приключенческих. Получается, что пользователя компьютера в работе с лабиринтами ожидают две основные задачи: генерация лабиринта и его прохождение.

Во время курсового проектирования необходимо разработать автоматизированную систему, с помощью которой можно генерировать лабиринт в автоматическом режиме с помощью специальных алгоритмов, а также находить выход из лабиринта вручную или используя алгоритмы нахождения пути.

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 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 [4,5].

1. Описание и анализ предметной области

Предметная область – 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 конкретной задачи [6].

Она включает объекты, изучаемые теорией, а также свойства, отношения и функции, которые принимаются во внимание в теории.

* 1. Описание предметной области
     1. Описание лабиринта

Лабиринт – структура, состоящая из запутанных путей, ведущих к выходу или в тупик [7].

С математической точки зрения лабиринт представляет собой топологическую задачу. Если у лабиринта имеется только один вход и необходимо найти дорогу к единственному выходу, то такую задачу всегда можно решить. Для этого достаточно, идя по лабиринту, все время одной рукой касаться стенки. Таким образом можно всегда найти выход из лабиринта, но путь не будет кратчайшим.

Тот же метод пригоден и в более традиционном случае, когда цель находится внутри лабиринта. Но только если в нем нет путей, по которым можно кружить вокруг цели и возвращаться в исходную точку. Иначе попасть внутрь "островка", вокруг которого проходит замкнутый маршрут, не удастся.

* + 1. Классификация лабиринтов

Лабиринты можно разбить по пяти различным классификациям: размерность, происхождение, топология, тесселяция и маршрутизация:

1. Размерность определяет количество измерений, в которых лабиринт находится [8]. Двумерный лабиринт расположен в двух измерениях и может быть легко отображен на бумаге, на рисунке 1 приведен пример такого лабиринта.

  
Рисунок – Двумерный лабиринт

Трехмерный лабиринт имеет несколько уровней, проходы могут подниматься вверх и опускаться вниз. Пример трехмерного лабиринта приведен на рисунке 2.

  
Рисунок 2 – Трехмерный лабиринт

Более высокие размерности лабиринтов трудно смоделировать в реальном мире, но они могут быть представлены математической абстракцией. Например, четырехмерный лабиринт может иметь порталы во времени, что будет интересным элементом компьютерной игры;

1. По происхождению лабиринты могут быть естественными и искусственными. Естественный лабиринт – это природный элемент, созданный либо стихией, либо животными, использующими лабиринт в качестве своего дома. Например, грызуны являются подземными жителями, использующими лабиринты. Они роют себе домики с несколькими выходами, чтобы в случае приближения хищника быстро скрыться из вида [9]. В роли тупиков в таких лабиринтах выступают «кладовые» комнаты и «детские». На рисунке 3 представлен план нор крота. Примерно так же выглядят норы хомяков, мышей, сурков и других мелких грызунов.

  
Рисунок 3 – План норы крота

Другим примером естественного лабиринта могут служить муравейники. У них тоже обычно несколько выходов наружу, тупиками являются хранилища ресурсов и комнаты с потомством. План муравейника в разрезе представлен на рисунке 4.

  
Рисунок – План муравейника

Искусственные лабиринты – лабиринты, созданные человекам в личных целях [9]. К ним можно отнести пещеры, созданные для добычи ресурсов. На рисунке 5 представлен план пещеры в деревне Борщово, которая в XIX веке была одним из основных источников кварца в Санкт-Петербургской губернии.

  
Рисунок 5 – План пещеры

Другим примером искусственных лабиринтов можно считать садовые лабиринты, созданные для украшения приусадебных участков, парков и площадей. На рисунке 6 показан лабиринт, построенный в 2016 году, для принца Уэльского в Дамфрис-хаусе;

1. Класс топологии описывает геометрию пространства, в котором расположен лабиринт. Бывают обычные лабиринты – лабиринты в эвклидовым пространстве, и необычные.

  
Рисунок 6 – Лабиринт в Дамфрис-хаусе

Примерами лабиринтов с необычной топологией могут быть лабиринты на поверхности куба, лабиринты на поверхности ленты Мёбиуса и лабиринты, эквивалентные находящимся на торе, где попарно соединены левая и правая, верхняя и нижняя стороны [8]. Пример лабиринта с необычной топологией представлен на рисунке 7;

  
Рисунок 7 – Пример лабиринта с необычной топологией

1. Тесселяция – классификация геометрии отдельных ячеек лабиринта. Существует огромное множество различных форм ячеек лабиринта, среди которых есть как стандартные формы: прямоугольные, треугольные, шестиугольные, так и необычные [8].

Дельта-лабиринты состоят из соединенных треугольников, при этом у каждой ячейки может быть до трех соединенных с ней проходов. Пример такого лабиринта представлен на рисунке 8.

  
Рисунок – Пример дельта-лабиринта

Гексагональные лабиринты составлены из шестиугольников, и у каждой ячейки может быть не более пяти проходов. Пример такого лабиринта представлен на рисунке 9.

  
Рисунок – Пример гексагонального лабиринта

Тета-лабиринты состоят из концентрических окружностей проходов, в которых начало или конец находится в центре, а другой - на внешнем крае. Пример такого лабиринта представлен на рисунке 10.

  
Рисунок – Пример тета-лабиринта

К необычной тесселяции можно отнести дзета-лабиринт, расположенный на прямоугольной сетке, но у которого помимо прямоугольных соединений ячеек, есть диагональные проходы под углом 45 градусов. Пример такого лабиринта представлен на рисунке 11.

  
Рисунок – Пример дзета-лабиринта

Фрактальный лабиринт – лабиринт, составленный из лабиринтов, в каждой его ячейке расположен лабиринт меньшего размера. Содержимое копирует себя, создавая бесконечно большой лабиринт. Пример такого лабиринта представлен на рисунке 12;

  
Рисунок – Пример фрактального лабиринта

1. Классификация по маршрутизации связана с типами проходов лабиринта в пределах его геометрии. Лабиринт с одиночным соединением, он же «идеальный» лабиринт – это лабиринт, в котором ровно один путь к любой другой точке. В нем нет петель, замкнутых цепей, недостижимых областей. Существует только одно решение такого лабиринта [8]. Пример «идеального» лабиринта приведен на рисунке 13.

  
Рисунок – Пример «идеального» лабиринта

Лабиринт с многократными соединениями или «плетеный» лабиринт – это лабиринт, в котором не тупиков. Он состоит из проходов, замыкающих и возвращающихся друг к другу, создавая петли вместо конечных точек. Пример приведен на рисунке 14.

  
Рисунок – Пример «плетеного» лабиринта

Одномаршрутный лабиринт – лабиринт без развилок, в котором есть только один длинный извивающийся проход, меняющий направление. Потеряться в таком лабиринте можно только от невнимательности или усталости. Пример одномаршрутного лабиринта приведен на рисунке 15.

  
Рисунок – Одномаршрутный лабиринт

* + 1. Описание алгоритмов генерации лабиринта

При создании лабиринтов требуется применять общие правила: не должно быть замкнутых контуров или петель, изолированных от других частей лабиринта областей, количество входов и выходов для неидеальных лабиринтов должно быть определено. Для генерации лабиринтов существуют множество алгоритмов [10]. Рассмотрим самые популярные из них:

* + - * алгоритм Эллера основан на построчной генерации, где между каждыми двумя клетками строки при определенных условиях случайным образом возникала стенка. Данный метод гарантирует отсутствие циклов и недоступных клеток посредством выполнения условий. В конце все клетки окажутся «в одном множестве», что будет означать, что между каждыми двумя клетками существует путь;
      * алгоритм на основе двоичных деревьев. Для каждой ячейки существует проход вверх или влево, но никогда не в обоих направлениях. В версии с добавлением стен для каждой вершины добавляется сегмент стены, ведущий вниз или вправо, но не в обоих направлениях. Каждая ячейка независима от всех других ячеек, за неимением нужды при её создании проверять состояние каких-то других ячеек;
      * алгоритм рекурсивного деления начинается с установления случайной горизонтальной или вертикальной стены, пересекающей доступную область в случайной строке или столбце. Потом вдоль нее случайно размещаются пустые места. Этот процесс рекурсивно повторяется для двух подобластей, сгенерированных разделяющей стеной. Для наилучших результатов нужно добавить отклонение в выборе горизонтали или вертикали на основе пропорций области. Например, область, ширина которой вдвое больше высоты, должна более часто делиться вертикальными стенами;
      * алгоритм выращивания леса отсортировывает множество ячеек случайным образом в список «новых». Одна или несколько ячеек перемещаются их из списка «новых» в список «активных». Далее выбирается ячейка из «активного» списка и вырезается проход в соседнюю несозданную ячейку из «нового» списка, добавляя новую ячейку в список «активных» и объединяя множества двух ячеек. Лабиринт завершён, когда список «активных» становится пустым;
      * алгоритм Краскала основан на том, что каждые ячейки помечаются уникальным идентификатором, а затем производится обход всех рёбер в случайном порядке. Если ячейки с обеих сторон от каждого ребра имеют разные идентификаторы, то удаляем стену и задаём всем ячейкам с одной стороны тот же идентификатор, что и ячейкам с другой. Если ячейки на обеих сторонах стены уже имеют одинаковый идентификатор, то между ними уже существует какой-то путь, поэтому стену можно оставить, чтобы не создавать петель.

В разрабатываемой системе будут использованы следующие алгоритмы:

* + - * алгоритм Олдос-Бродера основан на том, что в сетке выбирается точка и случайно перемещаемся в соседнюю ячейку. При условии, что осуществилось попадание в не вырезанную ячейку, в неё вырезается проход из предыдущей ячейки. Таким образом продолжается движение в соседние ячейки, пока во все ячейки не появится хотя бы один проход. Этот алгоритм создаёт лабиринты с низким показателем текучести;
      * алгоритм Эйлера основан на построчной генерации. Между каждыми двумя клетками строки при определенных условиях, чтобы не было циклов и недоступных клеток, случайным образом возникала стенка. При этом в конце все клетки окажутся «в одном множестве», что будет означать, что между каждыми двумя клетками существует путь.
    1. Описание алгоритмов нахождения пути

Существует множество способов решения лабиринтов, и каждый из них имеет собственные характеристики [10]. Рассмотрим некоторые из них.

* + - * cледование по одной руке основано на том, чтобы идти по проходам и при достижении развилки всегда поворачивать направо (или всегда налево). Чтобы применить такое решение лабиринта в реальном мире, нужно положить руку на правую (или левую) стену и постоянно держать её на стене в процессе прохождения лабиринта. При желании можно помечать уже посещённые ячейки и ячейки, посещённые дважды. В конце можно вернуться назад по решению, следуя только по ячейкам, посещённым один раз. Этот метод необязательно найдёт кратчайшее решение, и он совершенно не работает, если цель находится в центре лабиринта и его окружает замкнутая цепь;
      * алгоритм цепей начинается с указания нужных мест начала и конца, и тогда алгоритм всегда путь от начала до конца, если он существует. При этом решение склонно быть разумно коротким, если даже не кратчайшим. Это означает, что таким способом нельзя решать лабиринты, в которых неизвестно точное расположение конца;
      * волновой алгоритм основан на идее распространения волны от исходной точки в разные стороны. Начальное значение волны – ноль. То есть ближайшие точки, в которые можно пойти, например, верх, низ, левая и правая, и которые еще не затронуты волной, получают значение волны, а также некоторый модификатор проходимости этой точки. Чем он больше – тем медленнее преодоление данного участка. Значение волны увеличивается на один. Аналогично обрабатываются клетки, отходя от тех, на которой значение волны – 2. При этом на клетках с худшей проходимостью волна задержится. И так дальше все обрабатывается, пока не достигнута конечная точка маршрута. Сам путь в получившемся массиве значений волны вычисляется по наименьшим клеткам. Пример работы алгоритма приведен на рисунке 16.
  1. Описание систем-аналогов

Сравнительный анализ аналогичных систем – важный шаг в процессе разработки нового продукта.

  
Рисунок - Пример работы волнового алгоритма

Он позволяет не только понять сильные и слабые стороны конкурентов, но и определить направления для улучшений и внедрения новых идей. Такой анализ включает несколько ключевых аспектов:

* + - * изучение механик и структуры аналогичных лабиринтов для выявления интересных решений;
      * оценка визуальных элементов для анализа стилистических подходов, которые привлекают пользователей;
      * сравнение уровней взаимодействия с игроком и степени сложности в разных играх, что помогает определить оптимальные настройки для удержания интереса пользователей.
    1. Gotcha

На рынке видеоигр представлено колоссальное количество игр в жанре аркада для любителей интенсивного игрового процесса. Видеоигра «Gotcha» была разработана Алланом Олкорном, дизайнером Pong, а прототип был сконструирован Cyan Engineering, полунезависимой исследовательской дочерней компанией Atari. Игра внесла значительный вклад в жанр, будучи одной из первых игроков на рынке [11].

«Gotcha» – это игра в лабиринт для двух игроков, в которой один игрок пытается поймать другого. Лабиринт состоит из повторяющегося набора элементов, расположенных в несколько столбцов на экране. "Преследователь" представлен квадратом, а "Преследуемый" обозначен знаком плюс. По мере приближения Преследователя к преследуемому электронный звуковой сигнал воспроизводится с возрастающей частотой, пока Преследователь не достигнет Преследуемого.

На рисунке 17 приведена главная экранная форма программы «Gotcha», на которой продемонстрирован интерфейс игры.

  
Рисунок – Игровой процесс Gotcha

К достоинствам данной системы можно отнести постоянную генерацию нового лабиринта с целью усложнения игрового процесса для игроков.

К недостаткам системы относятся:

* + - * некоторые случайно сгенерированные лабиринты могут быть неравномерно сложными, что может привести к неравному опыту игры для разных игроков;
      * на этапе настроек не предусмотрено регулирование толщины стенок;
      * скромное оформление, которое со временем начинает приедаться глазу и надоедать уже во время игрового процесса.
    1. Онлайн генератор лабиринтов «Plottersvg»

В сети Интернет можно найти огромное количество реализаций генерации лабиринтов, которые могут значительно отличаться по функциональности и сложности. Эти реализации варьируются от простых веб-инструментов для создания лабиринтов до сложных программных решений и библиотек, используемых для научных исследований и разработки игр.

Некоторые системы предназначены для демонстрации работы алгоритмов генерации и решения лабиринтов, в то время как другие фокусируются на предоставлении пользователям гибких инструментов для создания уникальных лабиринтов для различных целей.

Рассмотрим один из таких инструментов на сайте [12]. Интерфейс программы, набор ее функций и основные настройки представлены на рисунке 18.

К достоинствам данной системы относятся:

* + - * регулирование толщины стенок и размера лабиринта по вертикали и горизонтали;
      * три режима расположения входов в лабиринт: по диагонали, слева и справа, сверху и снизу;

  
Рисунок – Генератор лабиринтов «Plottersvg»

* + - * возможность изменения цвета фона, пути и стенок лабиринта;
      * возможность сохранения и печати лабиринта.

К недостаткам системы относятся:

* + - * на сайте не представлена информация об алгоритмах, используемых при генерации лабиринта и нахождении пути;
      * нет возможности ручной расстановки входов и выходов;
      * не предусмотрено ручное прохождение и составление лабиринта.

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

* 1. Диаграмма объектов предметной области

Объектно-ориентированный анализ и проектирование (ООАП, Object-Oriented Analysis/Design) – технология разработки программных систем. В основу положена объектно-ориентированная методология для представления предметной области в виде объектов, которые являются экземплярами соответствующих классов.

Таблица 1 – Сравнительные характеристики систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название системы  Название показателя | «Gotcha» | Онлайн генератор лабиринтов «Plottersvg» | Разрабатываемая система |
| Возможность автоматической генерации лабиринта | + | + | + |
| Автоматическое прохождение лабиринта | + | + | + |
| Ручное прохождение лабиринта | – | – | + |
| Оформление по тематике | – | – | + |
| Выбор размера лабиринта | – | + | + |
| Визуализация прохождения лабиринта | + | – | + |
| Возможность выбора алгоритма генерации лабиринта | – | – | + |
| Возможность выбора алгоритма нахождения пути из лабиринта | – | – | + |

Общие принципы моделирования сложных систем и особенности процесса объектно-ориентированного анализа и проектирования ложатся в основы конструктивного использования языка UML.

Выбор выразительных средств для построения моделей сложных систем определяет и перечисляет те задачи, которые могут быть решены с использованием данных моделей.

Принцип абстрагирования является одним из основных принципов построения моделей сложных систем, который предписывает включать в модель только те аспекты проектируемой системы, которые имеют свое целевое предназначение к выполнению системой своих функций. При этом все второстепенные детали опускаются, чтобы чрезмерно не усложнять процесс анализа и исследования полученной модели. При таком подходе на финальном этапе получается модель, состоящая из разных объектов с более простым поведением, что в разы упрощает дальнейшую разработку [8].

Объект – элемент системы, в котором хранится перечень всех данных о себе и причисленных операций, производимых над ним. Обладает также основными свойствами объектно-ориентированного программирования.

Диаграмма объектов является статической составляющей взаимодействующих между собой объектов, она должна включить 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 перечислить свойства объектов. Диаграмма способна отразить нам пример того, как структуры данных будут выглядеть в определенный момент времени [9].

Лабиринт представляет собой совокупность следующих объектов: клетки, которые могут быть стеной или проходом, персонаж, который движется по проходам.

На рисунке 19 приведена диаграмма объектов предметной области.

  
Рисунок 19 – Диаграмма объектов предметной области

* 1. Постановка задачи

Во время курсового проектирования необходимо разработать автоматизированную систему генерирования структуры лабиринта различными способами, а также нахождение выхода из него в автоматическом режиме в соответствии с заданными алгоритмами. Система должна быть реализована в виде настольного приложения.

В системе должно быть реализовано две роли пользователей: администратор и игрок. Каждый пользователь должен авторизоваться в системе: ввести логин и пароль (длина логина должна быть от 6 до 16 символов, длина пароля – от 6 до 20 символов). Система должна проверить учетные записи и, в случае успеха, настроить интерфейс пользователя на заданную роль, в противном случае выдать сообщение об ошибке.

* + 1. Режим администратора

Основная задача администратора – создание лабиринта. Администратор должен предварительно задать параметры лабиринта: определить его размеры по горизонтали и вертикали (не менее 7 и не более 21), выбрать способ расстановки входа и выхода (автоматический или ручной), выбрать алгоритм генерации лабиринта (Олдоса-Бродера или Эйлера) и тему оформления. В системе должно быть реализовано 4 темы оформления (весна, лето, осень, зима). Лабиринт будет создаваться в автоматическом или ручном режиме и сохраняться в файл заданной структуры. Лабиринт должен быть толстостенным, вход и выход находятся на периметре лабиринта, но не в углах, а также вход и выход не совпадают друг с другом, стены должны образовывать тупики и пути, не должны образовывать изолированные части.

При этом система должна провести проверку корректности введённых параметров, и, в случае ошибки, выдать предупредительное сообщение с возможностью повторного ввода значений. В системе должен осуществляться контроль типов и диапазонов значений параметров. Система также должна обеспечивать контроль целостности лабиринта, предотвращая создание замкнутых или непроходимых участков.

* + 1. Режим игрока

Перед началом игры пользователь может загрузить один из предложенных лабиринтов из файла, а также выбрать тему оформления лабиринта. Пользователю с ролью игрок должен быть доступен выбор способа прохождения лабиринта (ручной и автоматический).

Если будет выбран автоматический режим прохождения лабиринта игроку будет предоставлен выбор алгоритма, с помощью которого нужно пройти лабиринт (одной руки или волновой). Помимо этого, будет дана возможность выбрать режим задержки перемещения персонажа, а также при автоматическом прохождении лабиринта будет присутствовать визуализация нахождения выхода из лабиринта.

Если будет выбран ручной режим прохождения лабиринта, тогда у пользователя должна быть возможность перемещения при помощи стандартных действий для нахождения выхода из лабиринта (вперёд, назад, вправо, влево). В программе предусмотрена возможность выдачи и просмотра справочной информации.

Таким образом, система должна решать следующие задачи:

1. функции системы:
   * аутентификация пользователя в системе, настройка интерфейса пользователя на заданную роль;
   * генерирование шаблона лабиринта по заданному размеру;
   * случайная расстановка входа и выхода;
   * контроль расстановки входа и выхода;
   * создание структуры лабиринта по заданному алгоритму;
   * проверка структуры лабиринта при загрузке;
   * визуализация генерации лабиринта;
   * визуализация прохождения лабиринта (видимая дорожка);
   * выдача справочной информации;
2. функции администратора:
   * авторизация пользователя в системе (ввод логина и пароля);
   * настройка параметров лабиринта при создании:
3. задание размера по горизонтали;
4. задание размера по вертикали;
5. выбор способа расстановки входа и выхода;
6. выбор алгоритма генерации лабиринта;
7. настройки темы оформления;
   * ручная расстановка входа и выхода;
   * сохранение лабиринта в файл заданной структуры;
   * просмотр лабиринта;
   * просмотр справочной информации;
8. функции игрока:
   * регистрация пользователя в системе (ввод логина и пароля);
   * авторизация пользователя в системе (ввод логина и пароля);
   * выбор лабиринта;
   * загрузка лабиринта из файла;
   * изменение темы оформления лабиринта;
   * выбор алгоритма прохождения лабиринта;
   * ручное прохождение лабиринта[[3]](#footnote-3);
   * выбор алгоритма прохождения лабиринта;
   * выбор режима прохождения лабиринта;
   * выбор задержки перемещения персонажа (для автоматического режима);
   * запуск алгоритма;
   * просмотр справочной информации.
9. Проектирование системы

Система – множество элементов, находящихся в отношениях или связях друг с другом, образующее целостность или органическое единство [15].

Проектирование – процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части. Результатом проектирования является проект – целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы [16].

Проектирование является частью стадии жизненного цикла системы, называемой определением системы. Результаты этой стадии являются входной информацией для стадии реализации системы.

Проектирование системы направлено на представление системы, соответствующее предусмотренной цели, принципам и замыслам; оно включает оценку и принятие решений по выбору таких компонентов системы, которые отвечают её архитектуре и укладываются в предписанные ограничения [16].

На этом этапе решаются задачи определения архитектуры приложения, его функций, требований к функционалу и интерфейсу, составляется структурная схема системы. Разрабатывается спецификация требований программного обеспечения (ПО), прототип интерфейса пользователя системы и информационно-логический проект системы, разрабатываются алгоритмы обработки данных.

* 1. Выбор и обоснование архитектуры системы

Архитектура – это базовая организация системы, которая описывает связи между компонентами этой системы и внешней средой, а также определяет принципы её проектирования и развития [17].

Существует множество архитектурных подходов к разработке ПО, рассмотрим наиболее распространенные из них:

1. монолитная архитектура. Это классический подход к разработке программного обеспечения. Все компоненты, модули и функции находятся в одном целом и работают в тесном взаимодействии друг с другом. Такой подход обеспечивает простоту и легкость в разработке, но может стать проблемой при масштабировании и поддержке продукта [18];
2. клиент-серверная архитектура. Этот подход разделяет программу на две основные части: клиентскую и серверную. Клиентская часть отвечает за интерфейс пользователя и взаимодействие с сервером, а серверная часть выполняет основные вычисления, отвечает за обработку и хранение данных. Такой подход позволяет эффективно использовать ресурсы и легко масштабировать систему [18];
3. распределенная архитектура. Такая архитектура предполагает, что компоненты программного обеспечения размещены на различных узлах сети, которые взаимодействуют друг с другом. Этот подход обеспечивает высокую отказоустойчивость, масштабируемость и производительность [18];
4. сервис-ориентированная архитектура. Это подход, при котором программное обеспечение разрабатывается в виде набора служб, которые могут быть использованы другими приложениями. Это позволяет создавать гибкие и масштабируемые системы, где каждая служба выполняет конкретную функцию и может быть использована повторно [18];
5. микросервисная архитектура. Это подход к разработке системы, при котором обеспечивается разделение ее компонентов на автономные составляющие – микросервисы. Они могут иметь собственную кодовую базу, обособленные базы данных, но при этом логически связаны друг с другом. Такая архитектура позволяет обновлять отдельные микросервисы без влияния на другие службы, что увеличивает гибкость и масштабируемость системы, но тем самым усложняет ее [19].

Разрабатываемая система будет обладать архитектурой монолитного настольного приложения по следующим причинам:

* популярность и распространенность подхода;
* высокая производительность, благодаря доступу к ресурсам машины, на которой установлено ПО, и архитектурному подходу;
* моментальная отзывчивость интерфейса программы к действиям пользователя;
* возможность работать в режиме оффлайн, что позволяет пользователям продолжать использовать приложение даже при отсутствии соединения с интернетом;
* повышенная безопасность данных, так как все необходимые файлы находятся исключительно на компьютере с ПО.
  1. Структурная схема системы

На этапе построения структурной схемы система разделяется по функциональному признаку на основные подсистемы, между которыми указываются информационные связи и связи по управлению, описывается основное назначение подсистем.

При разработке структурной схемы используется методология структурного проектирования, в основе которой лежит алгоритмическая декомпозиция и иерархия вида «часть-целое», учитывающая, что внутренние связи элементов внутри подсистем сильнее, чем связь между подсистемами. Декомпозиция системы может повторяться многократно, вплоть до уровня конкретных процедур, при этом должна быть обеспечена целостность системы, а все составляющие компоненты взаимоувязаны.

Структурная схема необходима для того, чтобы наглядно увидеть, как спроектирована система. На структурной схеме указывается связь между подсистемами.

На рисунке 20 приведена структурная схема разрабатываемой системы, в ее состав входят следующие подсистемы:

1. подсистема аутентификации, которая отвечает за регистрацию нового пользователя и проверку подлинности введенных данных существующим пользователем;

  
Рисунок 20 – Структурная схема системы

1. подсистема авторизации, которая отвечает за ввод учетных данных пользователя;
2. файловая система, которая отправляет запросы к файлам записи и получает ответы;
3. справочная подсистема, которая отвечает за выдачу сведений о системе и ее разработчиках;
4. подсистема визуализации, которая отвечает за графическое представление лабиринта и отображение выхода из него;
5. подсистема «Администратор», которая включает в себя следующие подсистемы:

* подсистему настройки параметров лабиринта при создании;
* подсистему создания лабиринта, которая включает в себя подсистему автоматической генерации лабиринта и подсистему ручной генерации лабиринта;

1. подсистема «Игрок», которая включает в себя следующие подсистемы:

* подсистему настроек параметров игры, которая отвечает за выбор способа прохождения и создания лабиринта, выбора темы оформления;
* подсистему нахождения пути, которая включает в себя подсистему автоматического нахождения пути и подсистему ручного нахождения пути.
  1. Разработка спецификации требований

Спецификация требований к программному обеспечению (Software Requirements Specification, SRS) – это описание программной системы, которую необходимо разработать. Она моделируется по образцу спецификации бизнес-требований. Спецификация требований к программному обеспечению устанавливает функциональные и нефункциональные требования, может включать набор вариантов использования пользователем. Правильная спецификация помогает предотвратить сбои программного обеспечения, получить четкое представление о разрабатываемом продукте [20].

Документ по программному обеспечению создает основу для соглашения между заказчиками и подрядчиками или поставщиками о том, как должен функционировать программный продукт (в рыночно ориентированном проекте эти роли могут играть отделы маркетинга и разработки) [20].

Спецификация требований к программному обеспечению – это строгая оценка требований перед более конкретными шагами проектирования системы. Ее цель – уменьшить количество последующих итераций, обеспечение реалистичной основы для оценки стоимости продукта, рисков и графиков. При правильном использовании спецификации требований к программному обеспечению могут помочь предотвратить провал программного проекта [20].

Документ спецификации требований к программному обеспечению содержит перечень требований, достаточных для разработки проекта. Чтобы вывести требования, разработчик должен иметь четкое и полное понимание разрабатываемых продуктов. Это достигается путем детального и непрерывного взаимодействия с проектной командой и заказчиком на протяжении всего процесса разработки программного обеспечения.

* + 1. Функциональная спецификация

Функциональные требования – это документ, который понятно и точно описывает существенные технические требования для объектов, материалов или операций. Спецификации помогают устранить дублирование и несоответствия, позволяют точно оценить необходимые действия и ресурсы, выступают в качестве согласующего и справочного документов о внесённых изменениях и предоставляют документацию с конфигурацией. Они дают точное представление о решении проблемы, повышая эффективность разработки системы и оценивая стоимость альтернативных путей проектирования, служат указанием для испытателей для верификации (качественной оценки) каждого технического требования [21].

Функциональная спецификация не определяет операции, происходящие внутри данной системы и каким образом будет реализована её функция. Вместо этого, она рассматривает взаимодействие с внешними агентами (например, персонал, использующий программное обеспечение; периферийные устройства компьютера или другие компьютеры), которые могут «следить», взаимодействуя с системой.

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 [22].

В таблице 3 приведен перечень исключительных ситуаций для разрабатываемой системы и описаны реакции системы на их возникновение.

Таблица 3 – 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 Файловая | 2.1 Попытка открытия файла с несобственным форматом | Выдача сообщения «Файл поврежден или недопустимого формата» |
| 2.2 Файл с заданным именем не существует | Выдача сообщения «Файл с заданным именем не существует» |

Таблица 2 – Перечень функций, выполняемых системой

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название  подсистемы | Название функции | Информационная среда | | | |
| Входные данные | | Выходные данные | |
| Назначение (наименование) | Тип, ограничения | Назначение (наименование) | Тип, ограничения |
|  |  |  |  |  |  |
| Настройки параметров игры | Выбрать режим прохождения | Список режимов прохождения | Ручной, автоматический | Текущий режим | Перечисление |
| Выбрать тему лабиринта | Список тем лабиринта | Зима, весна, лето, осень | Текущая тема | Перечисление |
| Выбрать алгоритм прохождения | Список алгоритмов | Волновой, одной руки | Текущий алгоритм прохождения | Перечисление |
| Выбрать скорость прохождения | Режимы скорости | Целое  0..4 | Текущая скорость | Целое |
| Настройки параметров лабиринта | Задать высоту лабиринта | Допустимый диапазон значений | Целое  7..21 | Текущая высота лабиринта | Целое |
| Код ошибки | Целое |
| Задать ширину лабиринта | Допустимый диапазон значений | Целое  7..21 | Текущая ширина лабиринта | Целое |
| Код ошибки | Целое |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Настройки параметров лабиринта | Создать шаблон лабиринта | Текущая высота | Целые | Лабиринт | Объект «Лабиринт» |
| Текущая ширина | Целые |
| Выбрать способ расстановки входа и выхода | Список вариантов расстановки | Ручной, автоматический | Текущий алгоритм расстановки | Целое |
| Поставить вход | Лабиринт | Объект | Координаты входа | Целые/нечетные, на периметре,  не в углах |
| Поставить выход | Лабиринт | Объект | Координаты выхода | Целые/нечетные, на периметре,  не в углах |
| Выбрать алгоритм генерации лабиринта | Список алгоритмов генерации | Эйлера,  Олдоса-Бродера | Текущий алгоритм генерации | Перечисление |
| Выбрать тему лабиринта | Список тем лабиринта | Зима, весна, лето, осень | Текущая тема | Перечисление |

* 1. Разработка прототипа интерфейса пользователя системы

Интерфейс – это набор инструментов, который позволяет пользователю взаимодействовать с программой. В более широком смысле термин обозначает любые инструменты для соприкосновения между разными системами и сущностями. Часто говорят, что графический интерфейс – это внешний вид сайта, программы или приложения [23].

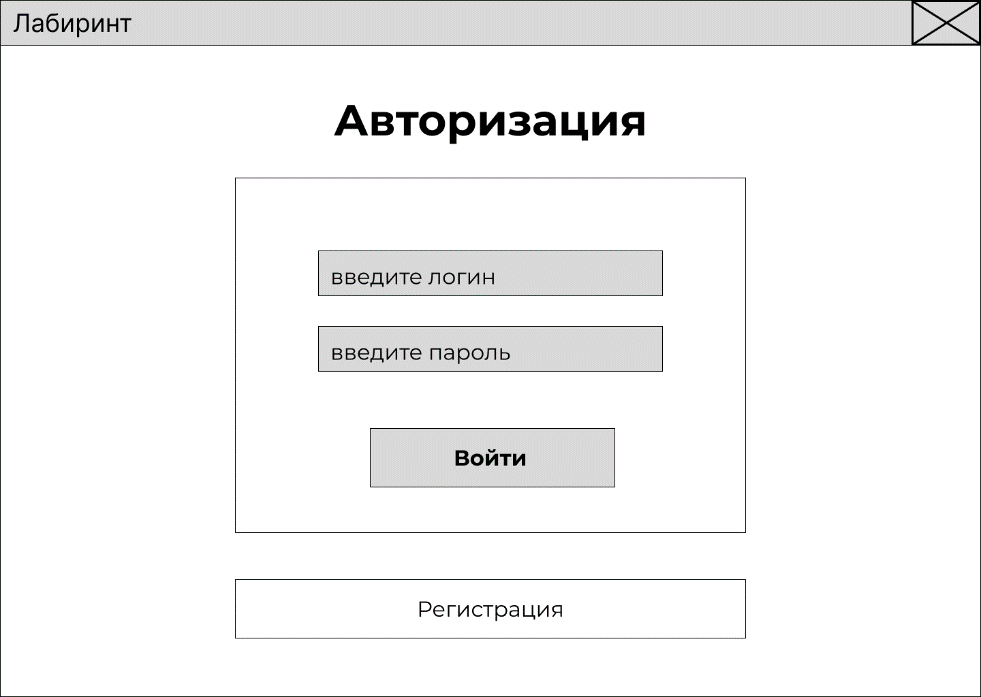
Интерфейс пользователя является одним из важнейших элементов программы, это та часть программы, которая находится у всех на виду. Недочеты в пользовательском интерфейсе могут серьезно испортить впечатление даже о самых многофункциональных программах.

Прототипирование интерфейсов представляет собой создание ранней, упрощенной версии сайта или приложения с целью тестирование интерфейса на предмет его комфортности и простоты использования для конечного пользователя до процесса разработки. Создание макетов позволяет не только понять, какие проблемы со стороны пользователя могут возникнуть в процессе использования продукта, но и предотвратить их еще на этапе планирования, уменьшить затраты на исправление ошибок еще до начала разработки [24].

Рассмотрим преимущества использования прототипирования:

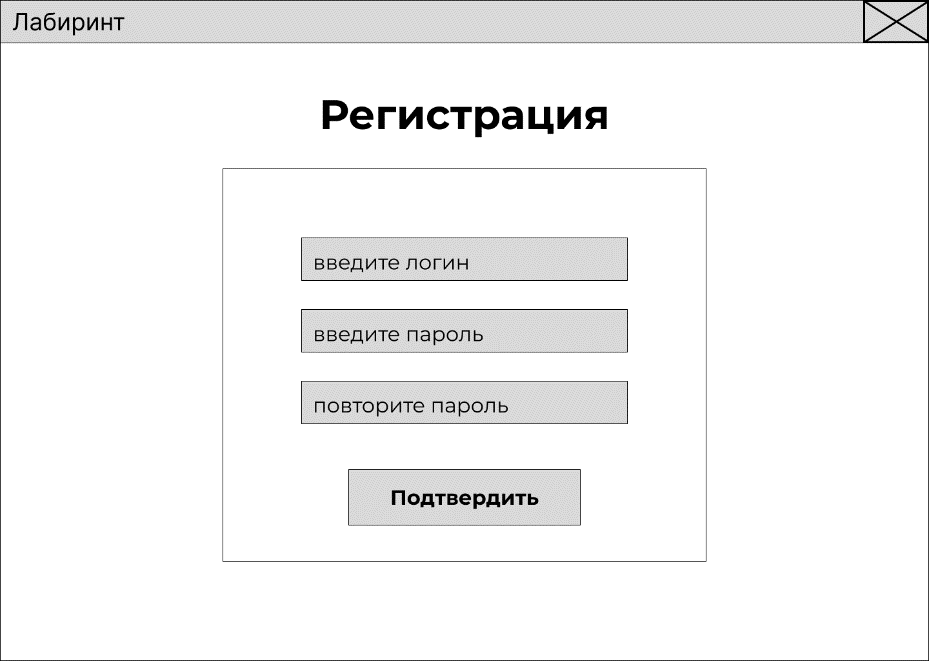
* проверка концепции: прототипирование может помочь увидеть, как система будет выглядеть и работать в реальности;
* исследование пользовательских потребностей: создание прототипа поможет понять, что пользователи хотят от системы и как эти потребности могут быть удовлетворены;
* тестирование: прототипирование может использоваться для тестирования продукта или системы на ранней стадии разработки, что может помочь выявить проблемы и недостатки до реализации системы;

На рисунке 21 приведен прототип экранной формы для авторизации пользователя.

  
Рисунок – Прототип экранной формы для авторизации

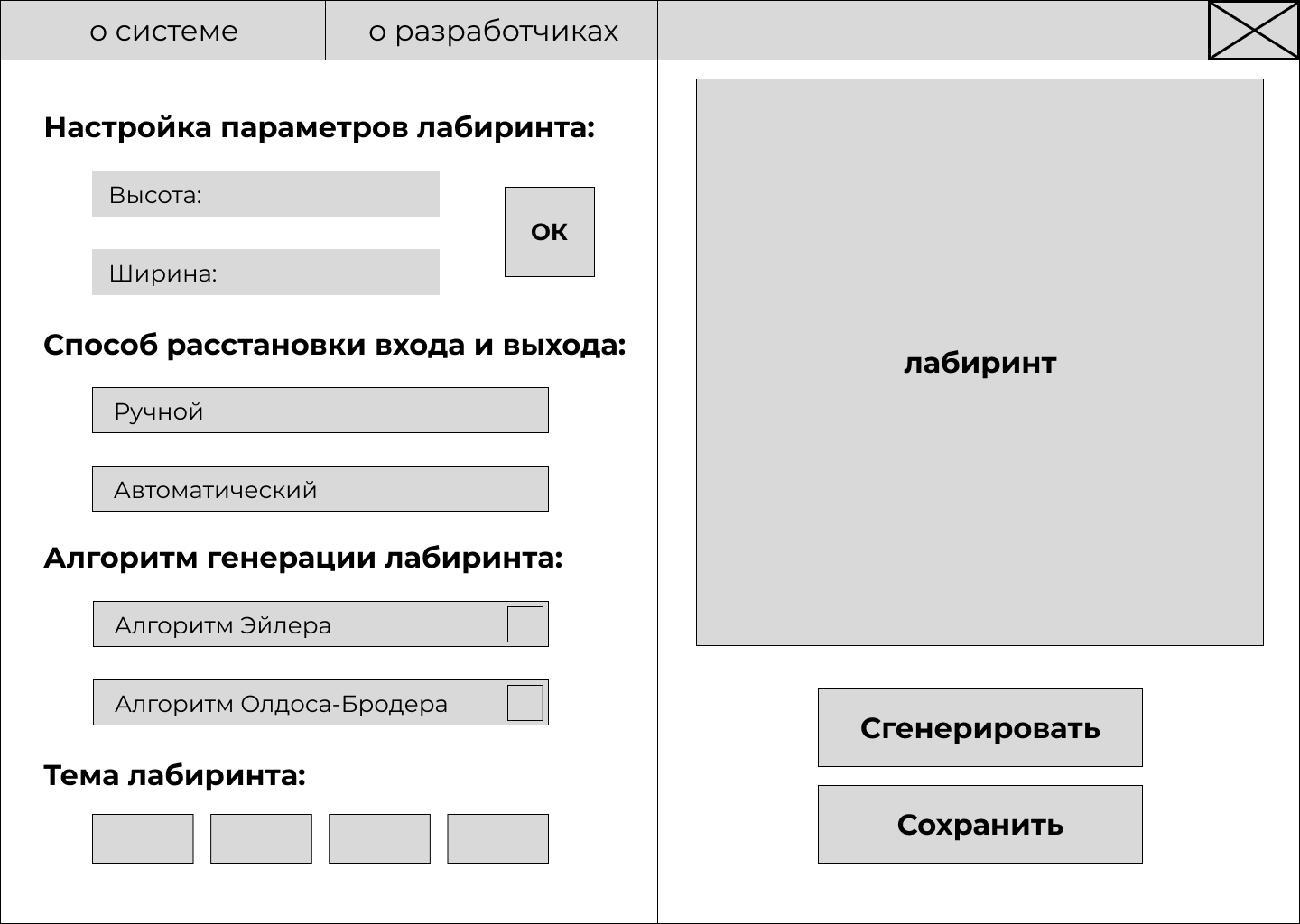
Пользователю необходимо будет ввести логин и пароль в соответствующие поля, если он уже был ранее зарегистрирован. Далее пользователь необходимо нажать кнопку «Войти» для перехода к основной форме. Если пользователь еще не зарегистрирован, то он сможет нажать на кнопку «Регистрация» для перехода к соответствующей форме

На рисунке 22 приведен прототип экранной формы для регистрации пользователей.

  
Рисунок – Прототип экранной формы для регистрации

Пользователю необходимо ввести логин и пароль в соответствующие поля. Также пользователю будет необходимо ввести пароль второй раз в специальное поле ввода для проверки совпадения паролей. Затем пользователь должен будет нажать кнопку «Подтвердить» для перехода к следующему экрану (форме «Авторизация»).

На рисунке 23 приведен прототип основной экранной формы для администратора.

  
Рисунок 23 – Прототип основной экранной формы для администратора

«О системе» и «О разработчиках» – кнопки, благодаря которым администратор сможет получить соответствующую информацию. На основном экране администратору предоставляется возможность настройки параметров лабиринта. В специально предназначенных полях администратор должен ввести высоту и ширину лабиринта и нажать на кнопку «ОК». Система должна сгенерировать шаблон лабиринта с установленными размерами, администратор должен выбирать способ расстановки входа и выхода (ручной и автоматический), алгоритм генерации лабиринта (Эйлера и Олдоса-Бродера), а также выбрать одну из предложенных на выбор тем (зима, весна, лето, осень). При выборе ручной расстановки входа и выхода, на сгенерированном шаблоне можно будет нажать на клетки, расположенные по периметру лабиринта, для выбора их в качестве входа и выхода в соответствующем порядке. После выбора алгоритма станет доступной кнопка «Сгенерировать». При нажатии на данную кнопку в правой части экрана будет отображен лабиринт с учетом всех введенных администратором параметров. Кнопка «Сохранить» дает возможность сохранить сгенерированный лабиринт в файл.

На рисунке 24 приведен прототип основной экранной формы для игрока.

  
Рисунок – Прототип основной экранной формы для игрока

«О системе» и «О разработчиках» – кнопки, благодаря которым игрок сможет получить соответствующую информацию. Кнопка «Загрузить» предназначена для загрузки лабиринта из файла. На основной экранной форме у игрока будет возможность настроить параметры игры: выбрать режим прохождения лабиринта (ручной и автоматический) и тему лабиринта. При выборе автоматического прохождения у игрока станет доступно кнопки выбора алгоритма прохождения лабиринта (волновой и одной руки). При автоматическом прохождении игрок сможет выбрать скорость перемещения (от 0 до 4). При нажатии на кнопку «Пройти» игрок сможет начать прохождение. Если выбрана скорость 0, то вместо кнопки «Пройти» появится кнопка «Сделать шаг» и алгоритм будет работать в пошаговом режиме.

На рисунке 25 приведена навигационная модель разрабатываемого приложения.

* 1. Разработка информационно-логического проекта системы

Информационно-логическая модель — это модель предметной области, которая определяет совокупность информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области. Она является схемой, описывающей взаимосвязи функциональных задач, решаемых на предприятии на всех этапах планирования, учёта и управления [26].

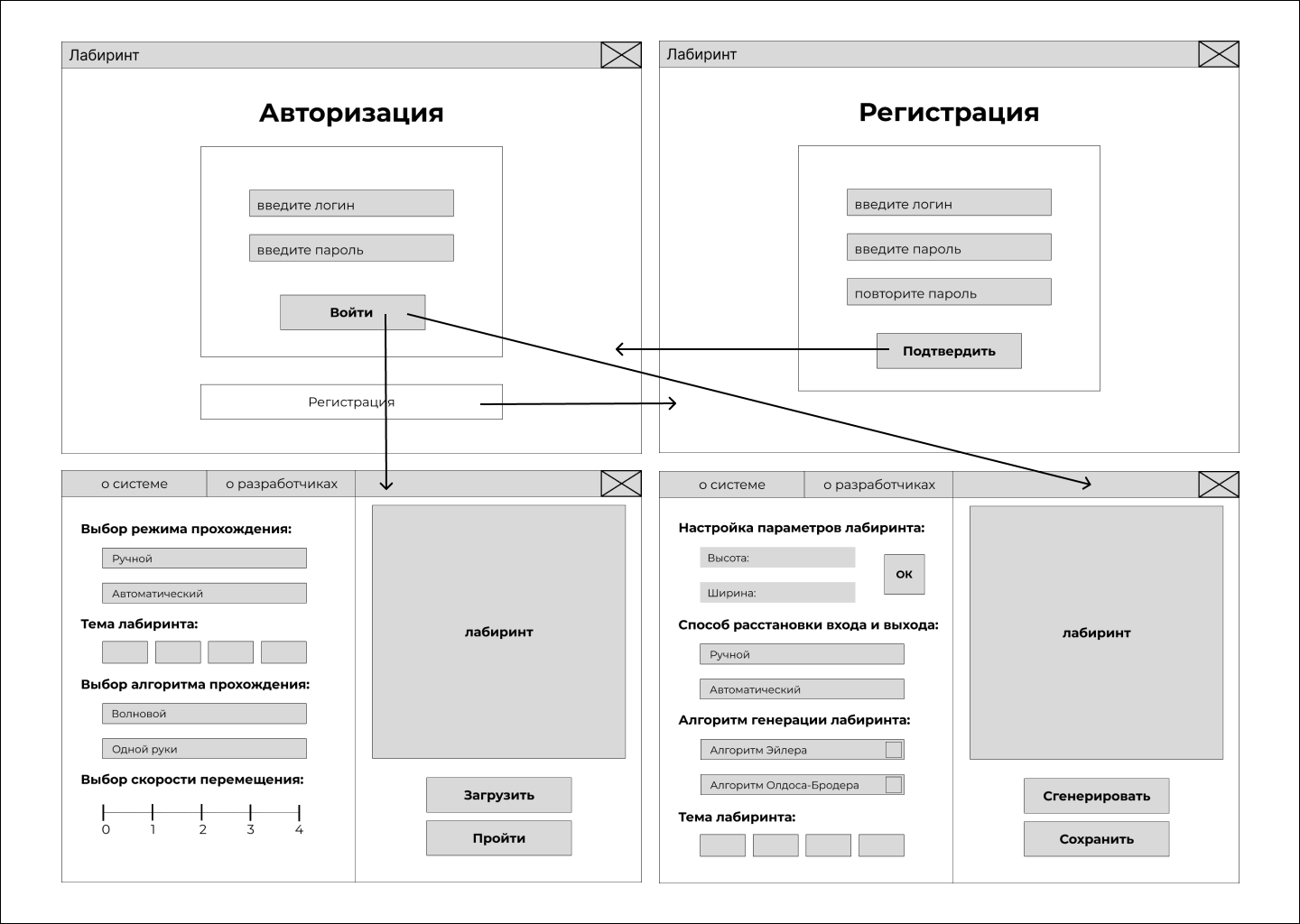
Эти модели используют графические представления, показывающие решения как исходной задачи, так и разрабатываемой системы. Как правило, графическое представление более понятно, чем описание требований на естественном языке [27].

Моделирование – это устоявшаяся и повсеместно принятая инженерная методика. Модели являются связующим звеном между процессом анализа и процессом проектирования системы [28].

* + 1. Язык UML

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

Унифицированный язык моделирования (Unified Modeling Language - UML) – это стандартный инструмент для разработки «чертежей» программного обеспечения. Его можно использовать для визуализации, спецификации, конструирования и документирования артефактов программных систем.

****  
Рисунок ‒ Навигационная модель приложения



Язык UML – это графический язык, описывающий с помощью диаграмм и схем разнообразные процессы и структуры. UML не является языком программирования, но на основании UML-моделей возможна генерация кода [29].

В нотации языка UML определены следующие виды канонических диаграмм:

* диаграмма вариантов использования (use-case diagram);
* диаграмма классов (class diagram);
* диаграмма кооперации (collaboration diagram);
* диаграмма последовательности (sequence diagram);
* диаграмма состояний (statechart diagram);
* диаграмма деятельности (activity diagram);
* диаграмма компонентов (component diagram);
* диаграмма развертывания (deployment diagram).

Рассмотрим для чего при разработке программных систем используют UML:

* для создания «чертежей» программы, схем, которые показывают, как будет устроено программное обеспечение изнутри. Это может быть описание связей между компонентами, модулей или сервисов, программных процессов и многого другого [30];
* для визуализации уже имеющейся программной структуры. Ряд инструментов позволяет создать UML на основе существующего кода, в таком случае диаграмма сгенерируется автоматически;
* для автоматической генерации кода или технической документации по нему, так как UML поддерживает возможность создания продукта на основе диаграмм;
* для внутренней и внешней коммуникации между сотрудниками, заказчиками и другими: картинки и диаграммы UML понятнее людям, чем текстовые описания.
  + 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.

Диаграмма вариантов использования создают, чтобы:

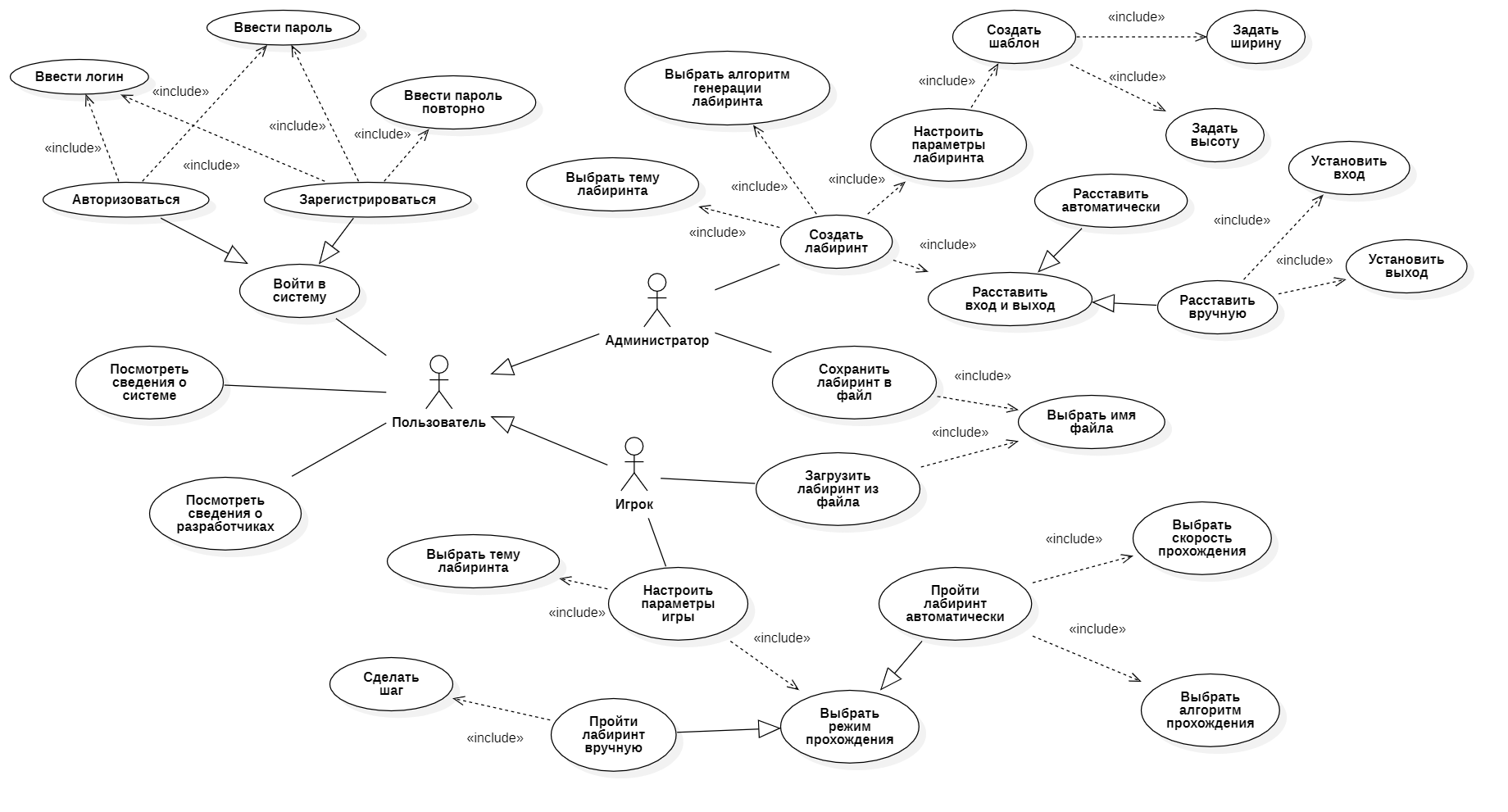
* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы [31];
* сформулировать общие требования к функциональному поведению проектируемой системы [31];
* разработать исходную концептуальную модель системы для её последующей детализации в форме логических и физических моделей [31];
* подготовить исходную документацию для взаимодействия разработчиков системы с её заказчиками и пользователями [31].

Основные части диаграммы вариантов использования:

* актер – это пользователь, который взаимодействует с системой. Это может быть человек, организация или даже сама система;
* варианты использования – овалы горизонтальной формы, обозначающие различные варианты использования, которые может иметь пользователь;
* ассоциация – линия между актёрами и вариантами использования;
* граничные рамки системы – блок, который устанавливает область действия системы для вариантов использования;
* отношения: включение, расширение и обобщение.

На рисунке 26 приведена диаграмма вариантов использования (пользователя).

Здесь должно быть описание диаграммы.

  
Рисунок – Диаграмма вариантов использования

* + 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.

* + 1. Сценарии определяются преподавателем
    2. 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 |
|  |  |
|  |  |
|  |  |
|  |  |

* + 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 приведена диаграммы последовательности системы для варианта 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 27 – 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

Вводные слова про необходимость разработки алгоритмов.

На рисунке ХХХ приведена схема алгоритма обработки элементов массива. Здесь должно быть краткое описание алгоритма.

  
Рисунок ХХХ – Схема алгоритма обработки элементов массива

На рисунке ХХХХ приведена схема алгоритма вычисления исходного выражения.

1

Начало

a

d

result = 0

push eax

eax = a

imul eax

add eax,1

eax = d

ebx = 4

Рисунок ХХХ – Схема алгоритма вычисления исходного выражения (начало)

Рисунок ХХХХ– Схема алгоритма вычисления исходного выражения (окончание)

sub eax, ebx

ebx = eax

eax = 24

pop ebx

idiv ebx

result

Конец

result = eax

idiv ebx

1

* 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 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 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 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

Разработка интерфейса важна по двум причинам:

* чтобы пользователи работали более продуктивно, программа должна быть простой в использовании;
* хороший интерфейс может стать преимуществом против конкурентов, плохой – послужить причиной неудачи всего проекта.

Основные особенности разработки интерфейса:

* простота: упрощение интерфейса до основных элементов поможет пользователю легко и быстро освоить продукт, простота способствует быстрому восприятию и снижает вероятность ошибок [24];
* интуитивность: интерфейс должен быть интуитивно понятным, чтобы пользователи сразу понимали, как с ним взаимодействовать без обучения [24];
* последовательность: элементы интерфейса должны быть единообразными: иметь одинаковый внешний вид и функционировать одинаково на всех экранах и во всех частях приложения [24];
* эстетика: интерфейс должен быть привлекательным, чтобы взаимодействие пользователя с системой было приятным, хороший визуальный дизайн повышает доверие пользователей к продукту и стимулирует их возвращаться к нему снова [23];
* скорость: система должна быстро реагировать на действия пользователя, оптимизация производительности интерфейса помогает снизить время ожидания и сделать взаимодействие плавным [23].

На рисунках приведены примеры того, как нужно оформить сведения о разработчиках.

  
Рисунок ХХХ – Сведения о разработчиках

  
Рисунок ХХХ – Сведения 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) [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.

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

Рисунок ХХХ – Диаграмма компонентов системы

Таблица ХХХ – Описание компонентов системы

|  |  |  |
| --- | --- | --- |
| Название компонента | Назначение компонента | Подсистема |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

* + 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.

На рисунке ХХХ приведена диаграмма развертывания системы. Здесь должно быть описание тех компонентов, которые развернуты на узлах ЭВМ.

Рисунок ХХХ – Диаграмма развертывания системы

* + 1. Диаграмма классов

В соответствии со спецификацией, приведенной в п. 2.5.6, и с учетом выбранного языка программирования (см. п. 2.8.1) 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 (Vсправки =0,8 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

Лабиринт [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Лабиринт (дата обращения: 17.09.2024).

Игровые лабиринты [Электронный ресурс]. URL: https://dskrnd.ru/blog/sovety-pokupatelyam/igrovye-labirinty-kakuyu-polzu-dlya-detskogo-razvitiya-oni-prinosyat/#:~:text=%Лабиринт%20–%20это%20не%20только%20интересное,игра%20способствует%20выплеску%20лишней%20энергии (дата обращения: 17.09.2024).

RAD (программирование) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/RAD\_(программирование) (дата обращения: 17.09.2024).

Краткая история UML [Электронный ресурс]. URL: http://techn.sstu.ru/kafedri/подразделения/1/MetMat/murashev/oop/lec/lec12.htm#:~:text=UML (дата обращения: 17.09.2024).

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

Технология объектно-ориентированного анализа и проектирование информационных систем [Электронный ресурс]. URL: https://textarchive.ru/c-1890952.html (дата обращения: 18.09.2024).

Объекто-ориентированный анализ и проектирование [Электронный ресурс]. URL: https://studopedia.ru/3\_33047\_ob-ektno-orientirovanniy-analiz-i-proektirovanie.html (дата обращения: 18.09.2024).

Лабиринты: классификация, генерирование, поиск решений [Электронный ресурс]. URL: https://habr.com/ru/articles/445378/ (дата обращения: 17.09.2024).

Лабиринт [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Лабиринт (дата обращения: 18.09.2024).

Алгоритмы генерации лабиринтов [Электронный ресурс]. URL: https://tproger.ru/articles/maze-generators?ysclid=m183grujpj942114810+,+ (дата обращения: 18.09.2024).

Gotcha [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Gotcha\_(video\_game) (дата обращения: 19.09.2024).

Plottersvg [Электронный ресурс]. URL: https://plottersvg.ru/maze-generator (дата обращения: 19.09.2024).

Принципы объектно-ориентированного проектирования [Электронный ресурс]. URL: https://studopedia.su/20\_118710\_printsipi-ob-ektno-orientirovannogo-proektirovaniya.html?ysclid=m18zcakru9268504810(дата обращения: 19.09.2024).

UML: обзор основных типов диаграмм, диаграмма объектов [Электронный ресурс]. URL: https://habr.com/ru/articles/800849/(дата обращения: 19.09.2024).

Системный анализ [Электронный ресурс]. URL: https://victor-safronov.ru/systems-analysis/lectures/rodionov/00.html (дата обращения: 29.09.2024).

Проектирование [Электронный ресурс]. URL: https://okbm.ru/proektirovanie/ (дата обращения: 19.09.2024).

Немного об архитектурах программного обеспечения [Электронный ресурс]. URL: https://habr.com/ru/  
companies/mws/articles/276297/ (дата обращения: 19.09.2024).

Виды архитектур ПО [Электронный ресурс]. URL: https://dzen.ru/a/ZdeO8pAHE1CaSJnz (дата обращения: 19.09.2024).

Что такое микросервисы и микросервисная архитектура? [Электронный ресурс]. URL: https://www.diasoft.ru/about/publications/20743/ (дата обращения: 19.09.2024).

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: 16.10.2024).

Класс Exception. Стандартные классы исключений. Операторы обработки исключений. [Электронный ресурс]. URL: https://saliyna.narod.ru/OOP/Lesson23/Lecture.html (дата обращения: 17.10.2024).

Интерфейс [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/interface/ (дата обращения: 07.10.2024).

Зачем нужно прототипирование интерфейсов? [Электронный ресурс]. URL: https://webdoka.ru/blog/zachem-nuzhno-prototipirovanie-ux/ (дата обращения: 20.10.2024).

Пользовательский интерфейс: правила и этапы разработки [Электронный ресурс]. URL: https://bangbangeducation.ru/point/ux-ui-dizain/polzovatelskij-interfejs/#5 (дата обращения: 07.10.2024).

Проектирование информационной системы [Электронный ресурс]. URL: https://3uch.ru/textbooks/qure/xio/qude (дата обращения: 27.10.2024).

1. Соммервиль, И. Инженерия программного обеспечения/ Иан Соммервиль. – М., СПб, Киев: Издательский дом «Вильямс», 2002. – 626 с. : ил.
2. ГОСТ 12.1.007-76. Система стандартов безопасности труда. Вредные вещества. Классификация и общие требования безопасности [Текст] – Дата введения 01.01.1977. М.: Стандартинформ, 2007. – 7 с.
3. Унифицированный язык моделирования UML [Электронный ресурс]. URL: https://samara.mgpu.ru/~dzhadzha/dis/15/200.html (дата обращения: 27.10.2024).
4. UML [Электронный ресурс]. URL: https://blog.skillfactory.ru/ glossary/uml/ (дата обращения: 27.10.2024).
5. Элементы графической нотации диаграммы вариантов использования [Электронный ресурс]. URL: https://intuit.ru/studies/courses/  
   32/32/lecture/1004 (дата обращения: 27.10.2024).

**Книги**

***Целиком***

.

*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: 20.12.2024).

Гидроэлектростанция (гидроэлектрическая станция, ГЭС) // Энциклопедический словарь юного техника М.: Издательство «Педагогика», 1987 [Электронный ресурс] // Библиотекарь.Ру: электрон. библ. 2006-2024. URL: http://www.bibliotekar.ru/enc-Tehnika/58.htm (дата обращения: 20.12.2024).

Субботин А.С. Основы гидротехники [Электронный ресурс]. URL: http://www.cawater-info.net/bk/dam-safety/files/subbotin.pdf (дата обращения: 03.02.2024).

Филиальная структура компании [Электронный ресурс] // Системный оператор Единой энергетической системы: [сайт]. [2009-2017]. URL: http://so-ups.ru/index.php?id=about (дата обращения: 20.12.2024).

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: 24.12.2024).

Автоматизированная система управления производственными процессами [Электронный ресурс] // MEScontrol: [сайт]. [2003-2017]. URL: http://mescontrol.ru/articles/systems (дата обращения: 02.04.2024).

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: 20.12.2024).

Пользовательский интерфейс [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: https://ru.wikipedia.org/ wiki/Пользовательский\_интерфейс (дата обращения: 17.03.2024).

***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: 17.10.2024).

**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.[[4]](#footnote-4)

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. Необязательно для реализации [↑](#footnote-ref-1)
2. *Количество страниц, рисунков, таблиц указывается с учетом приложений* [↑](#footnote-ref-2)
3. Необязательно для реализации [↑](#footnote-ref-3)
4. 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-4)