МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Институт «Информатика и вычислительная техника»

Кафедра «Автоматизированные системы обработки информации и управления»

Работа защищена с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Программная инженерия»

на тему «Автоматизация деятельности школы»

|  |  |
| --- | --- |
| Выполнили  студенты гр. Б23-784-1 | Старкова У. С.  Яновская Е. Б. |
| Принял  к.т.н., доцент каф. АСОИУ | Касимов Д. Р. |

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ижевск – 2024

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc185007092)

[ВВЕДЕНИЕ 3](#_Toc185007093)

[1 Выявление и анализ требований к разрабатываемому приложению по автоматизации деятельности школы 4](#_Toc185007094)

[1.1 Функциональные требования 4](#_Toc185007095)

[1.2 Нефункциональные требования 4](#_Toc185007096)

[1.3 Пользовательские истории 5](#_Toc185007097)

[1.4 Use-case диаграммы 5](#_Toc185007098)

[2 Планирование программного проекта 8](#_Toc185007099)

[2.1 Mindmap проекта 8](#_Toc185007100)

[2.2 Формирование команды 10](#_Toc185007101)

[2.3 Распределение задач на членов команды 11](#_Toc185007102)

[2.4 Приоритезация задач 12](#_Toc185007103)

[2.4.1 Высокоприоритетные задачи 12](#_Toc185007104)

[2.4.2 Среднеприоритетные задачи 13](#_Toc185007105)

[2.4.3 Низкоприоритетные задачи 14](#_Toc185007106)

[2.5 Распределение задач на спринты 15](#_Toc185007107)

[3 Проектирование программного продукта 19](#_Toc185007108)

[3.1 UML - диаграмма состояний 19](#_Toc185007109)

[3.2 UML - диаграмма последовательности 21](#_Toc185007110)

[3.3 UML - диаграмма классов 23](#_Toc185007111)

[4 Управление дефектами программного продукта 25](#_Toc185007112)

[4.1 Чек-лист для проверки 25](#_Toc185007113)

[4.2 Отчеты о дефектах (багах) 25](#_Toc185007114)

[4.3 Test-cases 28](#_Toc185007115)

[4.4 Структура проекта 30](#_Toc185007116)

[5 Оценка качества программного продукта 32](#_Toc185007117)

[ЗАКЛЮЧЕНИЕ 35](#_Toc185007118)

[СПИСОК ЛИТЕРАТУРЫ 36](#_Toc185007119)

ВВЕДЕНИЕ

В современном образовательном процессе важно использовать новые технологии, которые могут значительно упростить и оптимизировать деятельность школ, как для учеников, так и для преподавателей. Актуальность создания приложения по автоматизации деятельности школы вызвана ростом нагрузки на педагогов и необходимостью повышения качества учебного процесса. В условиях быстрого развития информационных технологий школы сталкиваются с проблемами, связанными с эффективной организацией учебного процесса, планированием расписания, ведением документации и взаимодействием с родителями. Эти факторы обуславливают необходимость создания эффективного и удобного инструмента, который поможет решить насущные проблемы и улучшить качество образования.

Цель данного приложения - создать удобное в использовании средство для обеспечения взаимодействия между всеми участниками образовательного процесса. Приложение будет включать в себя функции расписания, электронного журнала и электронного дневника, что позволит преподавателям и учащимся всегда оставаться в курсе изменений и упростит обучение.

Задачи, которые необходимо решить для достижения поставленной цели:

1. выявить и проанализировать требования к разрабатываемому приложению;
2. запланировать программный проект, а именно создать mindmap проекта, выявить задачи из требований, сформировать команду, распределить задачи на членов команды, приоритезировать задачи и разделить их на спринты;
3. спроектировать разрабатываемое приложение;
4. описать структуру проекта;
5. протестировать приложение;
6. оценить качество продукта.

Успешное выполнение этих задач позволит создать приложение, способное значительно упростить повседневные процессы в школе, сделать их более надежными и доступными для всех пользователей.

1 Выявление и анализ требований к разрабатываемому приложению по автоматизации деятельности школы

1.1 Функциональные требования

Первичные требования к приложению:

1. личный кабинет ученика и учителя;
2. электронный журнал;
3. расписание.

Основные функциональные требования к приложению.

1. Возможность зайти в личный кабинет по логину и паролю. Логин и пароль предоставляет администратор, их можно получить в школе. Логин и пароль генерируются администратором: логин - имя, фамилия из латинских символов и случайное число, пароль состоит из 8 строчных и прописных латинских букв и цифр.
2. Возможность учителя узнать расписание в электронном журнале.
3. Возможность ученика узнать расписание в электронном дневнике.
4. Возможность ученика просмотреть свою успеваемость в электронном дневнике.
5. Возможность ученика просмотреть свою домашние задания в электронном дневнике.
6. Возможность учителя отследить успеваемость класса в электронном журнале.
7. Возможность учителя отмечать посещаемость в электронном журнале.
8. Возможность учителя выставлять оценки в электронный журнал.
9. Возможность учителя добавлять комментарий к оценке.
10. Возможность учителя добавлять домашние задания в электронный журнал.

1.2 Нефункциональные требования

Нефункциональные требования.

1. На главном экране должны отображаться окна вариантов действий: расписание с предметами, кабинетами и временем занятий, электронный дневник – для ученика, электронный журнал – для учителя.
2. Для оптимальной работы программы необходимые минимальные требования операционной системы: Windows 7 и выше.
3. Поддерживаемые браузеры: Google Chrome 66.0 и более поздние.
4. Поддерживаемые браузеры: Microsoft Edge 25.0 и более поздние.
5. Поддерживаемые браузеры: Opera 45.0 и более поздние.
6. Поддерживаемые браузеры: Яндекс.Браузер 18.3.1 и более поздние.
7. Поддерживаемые браузеры: Safari 10.01.
8. Приложение должно быть доступно 99.9% времени в год.
9. Приложение в пике использования должно выдерживать до 10 000 обращений пользователей.

1.3 Пользовательские истории

Примеры пользовательских историй:

* я, как ученик, хочу узнать домашнее задание по пропущенному предмету;
* я, как администратор, хочу изменить расписание по запросу учителя;
* я, как учитель, хочу выставить оценки за урок и отметить отсутствующих учеников.

1.4 Use-case диаграммы

На рисунке 1.1 изображена Use-case диаграмма «Изменение расписания», в таблице 1.1 указано описание к этой диаграмме.

*Таблица 1.1* Use-case «Изменение расписания»

|  |  |
| --- | --- |
| **Название** | **Изменение расписания** |
| Описание | Администратор может изменить расписание на определенный день |
| Акторы | Администратор |
| Предусловия | Администратор получил запрос на изменение расписания (отменить урок) |
| Основной поток событий | 1. Администратор авторизовался в личном кабинете 2. Зашел в расписание 3. Выбрал необходимый класс 4. Выбрал нужный день для изменения 5. Выбрал необходимый урок 6. Удалил урок из расписания |
| Постусловия | Администратор успешно изменил расписание |
| Расширенные атрибуты | 1. Если администратор вводит несуществующую дисциплину, то система выдает ошибку.  2. Если администратор хочет предмет для удаления, то система запрашивает подтверждение на удаление |

Изображение выглядит как текст, диаграмма, круг, снимок экрана

Автоматически созданное описание

Рисунок 1.1 - Use-case диаграмма «Изменение расписания»

На рисунке 1.2 изображена Use-case диаграмма «Добавление комментария к оценке», в таблице 1.2 указано пояснение к данной диаграмме.

*Таблица 1.2* Use-case «Добавление комментария к оценке»

|  |  |
| --- | --- |
| **Название** | **Добавление комментария к оценке** |
| Описание | Учитель может добавить комментарий к оценке |
| Акторы | Учитель |
| Предусловия | Учитель захотел написать комментарий к оценке по своему предмету |
| Основной поток событий | 1. Учитель авторизовался в личном кабинете 2. Зашел в электронный журнал 3. Выбрал необходимый класс 4. Выбрал ученика 5. Выставил оценку 6. Добавил комментарий |
| Постусловия | Учитель успешно добавил комментарий к оценке |
| Расширенные атрибуты | 1. Если учитель, не добавил комментарий, то система запрашивает подтверждение |

Изображение выглядит как текст, диаграмма, Шрифт, круг

Автоматически созданное описание

Рисунок 1.2 - Use-case диаграмма «Добавление комментария к оценке»

На рисунке 1.3 представлена общая Use-case диаграмма программы.

Изображение выглядит как рисунок, диаграмма, текст, зарисовка

Автоматически созданное описание

Рисунок 1.3 – Use-case диаграмма

2 Планирование программного проекта

2.1 Mindmap проекта

На рисунке 2.1 представлена Mindmap проекта, по-другому ментальная карта. Она отображает ключевые функции приложения, направленного на автоматизацию различных аспектов школьной деятельности. Опишем ментальную карту.

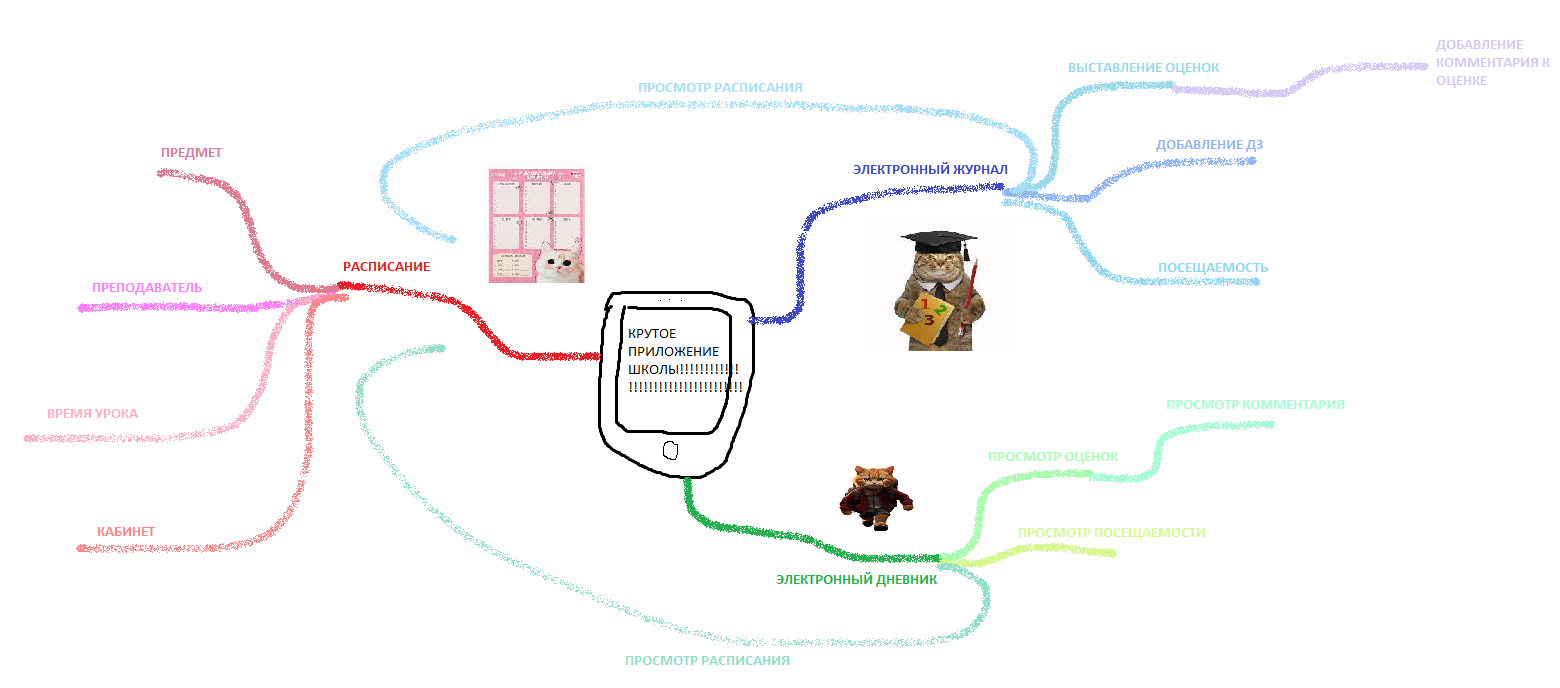
1. Электронный журнал:

* выставление оценок: автоматизированный процесс выставления оценок по различным предметам;
* просмотр расписания: удобный интерфейс для просмотра расписания занятий;
* добавление домашнего задания: возможность добавлять домашние задания к каждому предмету;
* посещаемость: отслеживание посещаемости учеников.

1. Электронный дневник:

* просмотр оценок;
* просмотр расписания;
* просмотр посещаемости.

1. От расписания идет ответвление на предмет, преподавателя, ведущего этот предмет, время урока и кабинет.

Рисунок 2.1 - Mindmap проекта

2.2 Формирование команды

Менеджер проекта (Middle).

Менеджер проекта — это ключевая фигура, отвечающая за руководство проектом и организацию всей работы над ним. В связи с размером команды и масштабом проекта, необходим специалист уровня middle. Нанимать senior специалиста нецелесообразно из-за сокращенной команды, в то время как junior не подойдет, так как требуется человек с определенным опытом. Его зарплата составит 715 руб/час

UI/UX дизайнер (Middle).

UI/UX дизайнер занимается созданием дизайна пользовательского интерфейса (UI) и пользовательского опыта (UX) приложения. Позиция middle предполагает наличие опыта работ, что соответствует требованиям нашего проекта, который не требует дизайна высокого уровня, как у senior. Зарплата дизайнера составит 595 руб/час.

Фронтенд-разработчик (Senior).

Фронтенд-разработчик отвечает за разработку и поддержку клиентской части приложения, то есть той, которая видна конечному пользователю. Он реализует дизайн пользовательского интерфейса, созданный дизайнером. Поскольку в команде всего один фронтенд-разработчик, требуется специалист уровня senior, чтобы он мог самостоятельно справляться со сложными задачами. Зарплата составит 1500 руб/час.

Бэкенд-разработчик (Senior и Middle).

Бэкенд-разработчик специализирован на разработке функциональной части веб-сайта или приложения. Его обязанности включают создание серверной части приложения, настройку базы данных и разработку API для взаимодействия с фронтенд-решениями. Уровень senior необходим для решения сложных задач, в то время как middle разработчик будет заниматься менее сложными задачами. Зарплата бэкенд-разработчина уровня senior составит 1715 руб/час, уровня middle - 1100 руб/час.

Тестировщик (Middle).

Тестировщик отвечает за проверку приложения на наличие ошибок и багов. В команде будет один тестировщик, который будет работать с веб-версией для браузера. Его зарплата - 650 руб/час.

Мы отказались от аналитика в команде, т.к. небольшой объем проекта подразумевает, что анализ требований можно выполнить на начальном этапе без отдельного аналитика и наше приложение не подразумевает сложной аналитики.

Лидирующий разработчик может выполнять функции, которые обычно возлагаются на тимлида. Это позволяет избежать дополнительных затрат.

2.3 Распределение задач на членов команды

Менеджер проекта Middle.

1. Руководить проектом и организовывать работу над ним.

UI/UX Дизайнер Middle.

1. Отрисовать макет входа в личный кабинет для учителя.
2. Отрисовать макет главного экрана для учителя.
3. Отрисовать макет входа в личный кабинет для ученика.
4. Отрисовать макет главного экрана для ученика.
5. Отрисовать макет расписания для ученика.
6. Отрисовать макет расписания для учителя.
7. Отрисовать электронный журнал.
8. Отрисовать электронный дневник.

Фронтенд-разработчик Senior.

1. Реализовать экран входа в личный кабинет для учителя согласно макетам.
2. Реализовать главный экран для учителя согласно макетам.
3. Реализовать вход в личный кабинет для ученика согласно макетам.
4. Реализовать главный экран для ученика согласно макетам.
5. Реализовать макет расписания для ученика.
6. Реализовать расписаниe для учителя.
7. Реализовать электронный журнал согласно макету.
8. Реализовать электронный дневник согласно макету.
9. Осуществить поддержку приложения браузером: Google Chrome 66.0 и более поздние.
10. Осуществить поддержку приложения браузером: Google Chrome 66.0 и более поздние.
11. Осуществить поддержку приложения браузером: Microsoft Edge 25.0 и более поздние.
12. Осуществить поддержку приложения браузером: Opera 45.0 и более поздние.
13. Осуществить поддержку приложения браузером: Яндекс.Браузер 18.3.1 и более поздние.
14. Осуществить поддержку приложения браузером: Safari 10.01 и более поздние.

Бэкенд-разработчик Senior.

1. Проработать структуру БД для хранения пользователей (администратор, учитель, ученик).
2. Проработать структуру БД для хранения расписания.
3. Проработать структуру БД для хранения электронного журнала.

Бэкенд-разработчик Мiddle.

1. Реализовать метод получения данных профиля текущего пользователя.
2. Реализовать микросервис, который на запрос фронт-приложения вернет расписание для конкретного пользователя.
3. Реализовать микросервис, который на запрос фронт-приложения вернет данные электронного журнала для учителя.
4. Реализовать микросервис, который на запрос фронт-приложения вернет данные электронного дневника.

Тестировщик Middle.

1. Протестировать приложение для браузера на наличие ошибок и багов: Google Chrome 66.0 и более поздние.
2. Протестировать приложение для браузера на наличие ошибок и багов: Google Chrome 66.0 и более поздние.
3. Протестировать приложение для браузера на наличие ошибок и багов: Microsoft Edge 25.0 и более поздние.
4. Протестировать приложение для браузера на наличие ошибок и багов: Opera 45.0 и более поздние.
5. Протестировать приложение для браузера на наличие ошибок и багов: Яндекс.Браузер 18.3.1 и более поздние.
6. Протестировать приложение для браузера на наличие ошибок и багов: Safari 10.01 и более поздние.

2.4 Приоритезация задач

2.4.1 Высокоприоритетные задачи

Менеджер проекта Middle.

1. Руководить проектом и организовывать работу над ним – 200 часов.

UI/UX Дизайнер Middle.

1. Отрисовать макет входа в личный кабинет для учителя – 5 часов.
2. Отрисовать макет главного экрана для учителя – 4 часа.
3. Отрисовать макет входа в личный кабинет для ученика – 5 часов.
4. Отрисовать макет главного экрана для ученика – 4 часа.
5. Отрисовать макет расписания для ученика – 7 часов.
6. Отрисовать макет расписания для учителя – 7 часов.
7. Отрисовать электронный журнал – 10 часов.

Фронтенд-разработчик Senior.

1. Реализовать экран входа в личный кабинет для учителя согласно макетам – 5 часов.
2. Реализовать главный экран для учителя согласно макетам – 4 часа.
3. Реализовать вход в личный кабинет для ученика согласно макетам – 5 часов.
4. Реализовать главный экран для ученика согласно макетам – 4 часа.
5. Реализовать макет расписания для ученика – 7 часов.
6. Реализовать расписаниe для учителя – 7 часов.
7. Реализовать электронный журнал согласно макету – 10 часов.

Бэкенд-разработчик Senior.

1. Проработать структуру БД для хранения пользователей (администратор, учитель, ученик) – 6 часов.
2. Проработать структуру БД для хранения расписания – 8 часов.
3. Проработать структуру БД для хранения электронного журнала – 8 часов.

Бэкенд-разработчик Мiddle.

1. Реализовать метод получения данных профиля текущего пользователя – 6 часов.
2. Реализовать микросервис, который на запрос фронт-приложения вернет расписание для конкретного пользователя – 6 часов.
3. Реализовать микросервис, который на запрос фронт-приложения вернет данные электронного журнала для учителя – 6 часов.

2.4.2 Среднеприоритетные задачи

UI/UX Дизайнер Middle.

1. Отрисовать электронный дневник – 10 часов.

Фронтенд-разработчик Senior.

1. Реализовать электронный дневник согласно макету – 10 часов.

Бэкенд-разработчик Мiddle.

1. Реализовать микросервис, который на запрос фронт-приложения вернет данные электронного дневника – 6 часов.

2.4.3 Низкоприоритетные задачи

Фронтенд-разработчик Senior.

1. Осуществить поддержку приложения браузером: Google Chrome 66.0 и более поздние – 5 часов.
2. Осуществить поддержку приложения браузером: Google Chrome 66.0 и более поздние – 5 часов.
3. Осуществить поддержку приложения браузером: Microsoft Edge 25.0 и более поздние – 5 часов.
4. Осуществить поддержку приложения браузером: Opera 45.0 и более поздние – 5 часов.
5. Осуществить поддержку приложения браузером: Яндекс.Браузер 18.3.1 и более поздние – 5 часов.
6. Осуществить поддержку приложения браузером: Safari 10.01 и более поздние – 5 часов.

Тестировщик Middle.

1. Протестировать приложение для браузера на наличие ошибок и багов: Google Chrome 66.0 и более поздние – 5 часов.
2. Протестировать приложение для браузера на наличие ошибок и багов: Google Chrome 66.0 и более поздние – 5 часов.
3. Протестировать приложение для браузера на наличие ошибок и багов: Microsoft Edge 25.0 и более поздние – 5 часов.
4. Протестировать приложение для браузера на наличие ошибок и багов: Opera 45.0 и более поздние – 5 часов.
5. Протестировать приложение для браузера на наличие ошибок и багов: Яндекс.Браузер 18.3.1 и более поздние – 5 часов.
6. Протестировать приложение для браузера на наличие ошибок и багов: Safari 10.01 и более поздние – 5 часов.

2.5 Распределение задач на спринты

Спринт — это короткий временной интервал, в течение которого scrum-команда выполняет заданный объем работы. Спринты лежат в основе методологий scrum и agile, и правильный выбор спринтов поможет вашей agile‑команде выпускать более качественное программное обеспечение без лишней головной боли.

«При использовании Scrum продукт разрабатывается в ходе нескольких итераций с фиксированной продолжительностью, которые называются спринтами и разбивают большие сложные проекты на небольшие задачи», — говорит Меган Кук, директор по продукту Jira в Atlassian [2].

Мы посчитали количество часов для каждой задачи и в сумме вышло 200 часов. Далее мы поделили задачи на спринты по неделе (рис. 2.2, 2.3 и 2.4). Распараллелив задачи на спринты, мы поняли, что на выполнение проекта понадобиться 2 недели и 2 дня, а это 90 часов.

Стоимость проекта составила 369820 рублей.

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

Автоматически созданное описаниеРисунок 2.2 – 1 неделя спринта

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2.3 – 2 неделя спринта

Изображение выглядит как текст, снимок экрана, Шрифт, Параллельный

Автоматически созданное описание

Рисунок 2.4 – 3 неделя спринта

3 Проектирование программного продукта

3.1 UML - диаграмма состояний

На рисунке 3.1 представлена диаграмма состояний, которая изображает процесс выставления оценки. Она построена с помощью инструмента draw.io [1].

Начальное состояние – «Оценка не выставлена». Процесс запускает Trigger: «Пользователь хочет выставить оценку».

Состояние «Оценка в процессе выставления»:

* входное действие (entry): «Запрашивается оценка, которую нужно выставить»;
* действие (do): «Проверка оценки на валидность».

Переходы из состояния «Оценка в процессе выставления»:

* если оценка прошла проверку, происходит переход к состоянию «Оценка выставлена» (Trigger: «Оценка прошла проверку»);
* если оценка не прошла проверку, происходит переход к состоянию «Оценка отклонена» (Trigger: «Оценка не прошла проверку»).

Состояние «Оценка выставлена»:

* выходное действие (exit): уведомление «оценка выставлена».

Состояние «Оценка отклонена»:

* выходное действие (exit): уведомление "Оценка не подходит".

Конечное состояние:

* процесс завершается, когда оценка либо успешно выставлена, либо отклонена.

Эта диаграмма показывает, как система обрабатывает выставление оценки, начиная с запроса пользователя и заканчивая либо успешным завершением, либо отклонением оценки.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 3.1 – Диаграмма состояний

3.2 UML - диаграмма последовательности

На рисунке 3.2 представлена диаграмма последовательности, описывающая процесс изменения расписания, включая взаимодействие между различными участниками и системами. Основные шаги процесса:

1. Инициатор процесса учитель: он запрашивает изменение расписания у администратора.
2. Обработка запроса администратором.
3. После обработки запроса изменения сохраняются в базе данных.
4. Система расписания обрабатывает запрос и вносит изменения в расписание.
5. Расписание отправляет сигнал об изменении в электронный журнал, электронный журнал обновляется и отображает обновленное расписание.
6. Микросервис уведомлений получает сигнал об изменении и уведомляет ученика о новом расписании.

Диаграмма наглядно показывает последовательность взаимодействий между следующими элементами: учитель, администратор, база данных, расписание, электронный журнал, микросервис уведомлений и ученик.

Изображение выглядит как снимок экрана, диаграмма, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.2 – Диаграмма последовательности

3.3 UML - диаграмма классов

На рисунке 3.3 показана диаграмма классов. Она описывает модель взаимодействий в образовательной системе, где есть различные роли пользователей и функциональные компоненты системы.

Основные элементы диаграммы:

1. Класс-родитель «Пользователь». Он содержит поля логин, фамилию, имя и отчество. От него наследуются классы «Ученик», «Администратор» и «Учитель».
2. Класс-наследник «Ученик». Он содержит поле с номером телефона родителя и методы просмотра расписания и данных учителя.
3. Класс-наследник «Администратор» имеет методы редактирования расписания, электронного журнала и электронного дневника.
4. Класс-наследник «Учитель» содержит поля с электронной почтой и номером телефона. Он имеет методы выставления оценки, добавления комментария и домашнего задания.
5. Классы функциональных компонентов системы:

* расписание имеет методы возвращения расписания за разные интервалы времени. Этот класс зависит от администратора, т. к. он может его редактировать;
* электронный журнал имеет методы возвращения оценки и комментария. Он зависит от расписания.
* электронный дневник является наследником электронного журнала.

Диаграмма показывает, какие действия могут выполнять разные классы в системе и как они взаимодействуют с ключевыми функциями, такими как расписание, электронный журнал и дневник.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 3.3 – Диаграмма классов

4 Управление дефектами программного продукта

4.1 Чек-лист для проверки

Сформируем чек-лист для тестирования из требований.

1. Проверить возможность зайти в личный кабинет по логину и паролю. Логин и пароль генерируются администратором: логин - имя, фамилия из латинских символов и случайное число, пароль состоит из 8 строчных и прописных латинских букв и цифр.
2. Проверить возможность учителя узнать расписание предметов.
3. Проверить возможность ученика узнать расписание предметов.
4. Проверить возможность ученика просмотреть свою успеваемость.
5. Проверить возможность ученика просмотреть домашние задания в электронном дневнике.
6. Проверить возможность учителя отследить успеваемость класса в электронном журнале.
7. Проверить возможность учителя отмечать посещаемость.
8. Проверить возможность учителя выставлять оценки.
9. Проверить возможность учителя добавлять комментарий к оценке.
10. Проверить возможность учителя добавлять домашние задания.
    1. Отчеты о дефектах (багах)

Критический баг (critical) — существование дефекта приводит к масштабным последствиям катастрофического характера, например: потеря данных, раскрытие конфиденциальной информации, нарушение ключевой функциональности приложения и т.д [3]. Пример критического бага представлен в таблице 4.1.

*Таблица 4.1* – Критический баг

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Иденти-фикатор** | **Краткое описание** | | **Подробное описание** | | | **Шаги по воспроизведению** | | |
| 1 | Сбой в системе аутентификации | | Пользователи не могут войти в систему, что полностью блокирует доступ к электронному журналу. | | | 1.Ввод логина  2.Ввод пароля  3.Попытаться войти | | |
| **Воспроизводимость** | | **Важность** | | **Срочность** | **Симптом** | | **Возможность обойти** | **Комментарий** |
| Иногда | | Критическая | | Наивысшая | Крах системы | | Нет | Бесконечная загрузка при попытке войти |

Высокий по важности баг (major) — существование дефекта приносит ощутимые неудобства многим пользователям в рамках их типичной деятельности, например: недоступность вставки из буфера обмена, неработоспособность общепринятых клавиатурных комбинаций, необходимость перезапуска приложения при выполнении типичных сценариев работы [3]. Пример высокоприоритетного бага представлен в таблице 4.1.

*Таблица 4.2* – Высокоприоритетный баг

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Иденти-фикатор** | **Краткое описание** | | **Подробное описание** | | | **Шаги по воспроизведению** | |
| 2 | Проблемы с уведомлениями | | Уведомления о новых заданиях или изменениях в расписании не приходят пользователям, что приводит к пропуску важной информации | | | 1.Авторизоваться в приложении  2.Перейти в настройки уведомлений  3.Включить уведомления  4.Создать событие, требующее уведомление | |
| **Воспроизводимость** | | **Важность** | | **Срочность** | **Симптом** | **Возможность обойти** | **Комментарий** |
| Иногда | | Высоко-приоритетная | | Обычная | Нереализованная функциональность | Нет | Уведомления вовсе не приходят |

Низкий по важности баг (minor) — существование дефекта редко обнаруживается незначительным процентом пользователей и (почти) не влияет на их работу, например: опечатка в глубоко вложенном пункте меню настроек, некое окно сразу при отображении расположено неудобно (нужно перетянуть его в удобное место), неточно отображается время до завершения операции копирования файлов [3]. Пример низкоприоритетного бага представлен в таблице 4.2.

*Таблица 4.2* – Низкоприоритетный баг

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Иденти-фикатор** | **Краткое описание** | | **Подробное описание** | | | **Шаги по воспроизведению** | |
| 3 | Малозначительная опечатка в тексте на главном экране. «Электроный журнал» написан с одной н | | В системе имеются небольшие грамматические ошибки, которые не влияют на функциональность, но портят общее впечатление | | | 1.Войти в приложение  2.Зайти на главный экран  3.Обнаружить ошибку в написании «Электронного журнала» | |
| **Воспроизводимость** | | **Важность** | | **Срочность** | **Симптом** | **Возможность обойти** | **Комментарий** |
| Всегда | | Низко-  приоритетная | | Обычная | Косметический дефект | Нет | Визуально заметный недостаток интерфейса, не влияющий на функциональность приложения |

4.3 Test-cases

Позитивные, или положительные тест-кейсы проверяют, что система адекватно реагирует на корректные данные. Например, если при регистрации ввести в поле логина существующий, корректно написанный email, ещё не зарегистрированный в системе, сайт поймёт это правильно и допустит регистрацию [4]. Пример позитивного тест-кейса представлен в таблице 4.4.

*Таблица 4.4* - «Проверка возможности учителя отметить отсутствие ученика по болезни»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Идентифи-катор** | **Приоритет** | **Модуль и подмодуль приложения** | **Шаги тест-кейса** | **Ожидаемый результат по каждому шагу тест-кейса** |
| 1 | Высокий | Электронный журнал | 1.Выбрать вкладку «Электронный журнал»  2.Выбрать класс  3.Нажать пкм на ячейку смежную с учеником и датой  4.Из всплывшего меню выбрать вариант «Отсутствовал по болезни»  5. Нажать Enter для подтверждения | 1.Вкладка «Электронный журнал» открывается  2.Открывается список выбранного класса  3.Всплывает с меню с выбором вида посещения  4.Выбранный вид посещаемости появляется в списке  5.Всплывает уведомление «Сохранено)» |

Негативные, или отрицательные тест-кейсы показывают, что система умеет работать с некорректными данными. Например, если не написать в email значок @ или пропустить точку, сайт сообщит об ошибке и не допустит регистрацию [4]. Пример негативного тест-кейса представлен в таблице 4.5.

*Таблица 4.5* - «Проверка возможности учителя выставить невалидную оценку»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Идентифи-катор** | **Приоритет** | **Модуль и подмодуль приложения** | **Шаги тест-кейса** | **Ожидаемый результат по каждому шагу тест-кейса** |
| 2 | Высокий | Журнал | 1.Выбрать вкладку «Электронный журнал»  2.Выбрать класс  3.Нажать лкм на ячейку смежную с учеником и датой  4. Напечатать оценку 6  5. Нажать Enter для подтверждения | 1.Вкладка «Электронный журнал» открывается  2.Открывается список выбранного класса  3.Ячейка выделяется и появляется курсор  4.Оценка 6 отображается в ячейке  5.Всплывает уведомление «Неправильно введенная оценка. Попробуйте снова!» |

Деструктивные тест-кейсы служат для проверки прочности системы. Например, позволяют убедиться, что в поле для email нельзя ввести команду, которая удалит базу данных зарегистрированных пользователей [4].

*Таблица 4.6* - «Проверить возможность учителя добавлять комментарий к оценке»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Идентифи-катор** | **Приоритет** | **Модуль и подмодуль приложения** | **Шаги тест-кейса** | **Ожидаемый результат по каждому шагу тест-кейса** |
| 3 | Высокий | Журнал | 1.Выбрать вкладку «Электронный журнал»  2.Выбрать класс  3.Нажать на ячейку с оценкой пкм  4.Выбрать «Добавление комментария»  5.Начать писать комментарий  6.Устройство разрядилось и выключилось | 1.Вкладка «Электронный журнал» открывается  2.Открывается список выбранного класса  3.Всплывает меню с выбором «Добавить комментарий»  4.Появляется окно с курсором, куда нужно вписать комментарий  5.Вводится комментарий  6.Комментарий не сохранился, но закэшировался системой. Можно вернуться и продолжить дальше |

4.4 Структура проекта

Опишем структуру нашего проекта в Visual Studio, которая представлена на рисунке 4.1: в проекте созданы папки Models и Application. Папка Models содержит в себе классы, описанные в диаграмме классов из лабораторной работы №6. Папка Application содержит различные классы-сервисы, например сервис по добавлению домашнего задания.

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

Автоматически созданное описание

Рисунок 4.1

Ниже на рисунках 4.2 и 4.3 представлены скриншоты с ветками и историей коммитов из репозитория GitHub. Каждая лабораторная работа расположена в своей папке и начинается с новой ветки. В комментариях к коммитам указаны изменения, которые были внесены.

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 4.2

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

Автоматически созданное описание

Рисунок 4.3

5 Оценка качества программного продукта

Оценим качество нашего программного продукта по 5-бальной шкале, где 1 – самая низкая оценка, а 5 – самая высокая. Занесем оценки в таблицу 5.1.

*Таблица 5.1* – Оценка качества

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **Описание свойств показателя** | **Оценка** | **Пояснение** |
| Функциональность | Группа свойств ПО, обуславливающая его способность выполнять определенный перечень функций, которые удовлетворяют потребностям в соответствии с назначением | 4 | Полнота функций – 4  Точность – 5  Интероперабельность – 3  Защищенность – 4  Приложение обеспечивает выполнение необходимых функций для автоматизации школьной деятельности, включая электронный журнал, электронный дневник и расписание. Все основные функции реализованы, но есть возможности для улучшения, например, добавления функций новостей и мессенджера. |
| Надежность | Группа свойств, обусловливающая способность ПО сохранять работоспособность и преобразовывать исходные данные в результат за установленный период времени, характер отказов которого является следствием внутренних дефектов и условий его применения | 3 | Завершенность – 3  Отказоустойчивость – 3  Восстанавливаемость – 3  Согласованность – 4  Приложение стабильно работает при стандартных условиях использования. Однако при интенсивной нагрузке (например, во время большого количества запросов) могут возникать незначительные задержки или сбои. |
| Удобство применения | Совокупность свойств ПО для предполагаемого круга пользователей и отражающих легкость его освоения и адаптации к изменяющимся условиям эксплуатации, стабильность работы и подготовки данных, понимаемость результатов, удобства внесения изменений в программную документацию и в программы | 5 | Понимаемость – 5  Обучаемость – 5  Привлекательность – 5  Согласованность – 5  Интерфейс приложения интуитивно понятен для любых пользователей (ученик, учитель, администратор) и визуально приятен. В нем сходу легко разобраться и смело пользоваться. |
| Сопровождаемость | Группа свойств, определяющая усилия, необходимые для выполнения, приспособленность к диагностике отказов и последствий внесения изменений, модификации и аттестации модифицируемого ПО | 4 | Анализируемость – 4  Изменяемость – 4  Стабильность – 4  Тестируемость – 5  Согласованность – 3  Приложение легко поддерживается разработчиками: обновления выпускаются своевременно, а диагностика и устранение ошибок не требуют значительных усилий. Но есть куда расти, так как команда разработчиков не такая большая и иногда не хватает рук для решения ошибок. |
| Рациональность | Группа свойств, характеризующаяся степенью соответствия используемых ресурсов среды функционирования уровню качества (надежности) функционирования ПО при заданных условиях применения | 4 | Реактивность – 5  Используемость ресурсов – 4  Согласованность – 4  Приложение эффективно использует ресурсы компьютеров и серверов. Время отклика и загрузки страниц оптимально, но при работе на устаревшем оборудовании производительность может немного снижаться. |
| Переносимость | Группа свойств ПО, обеспечивающая его приспособленность для переноса из одной среды функционирования в другие, усилия для переноса и адаптацию ПО к новой среде функционирования | 3 | Адаптируемость – 2  Простота настройки – 4  Совместимость – 3  Заменяемость – 4  Согласованность – 3  Приложение поддерживается только на Windows и на нескольких самых популярных браузерах, поэтому приспособленность слабо реализована и потребуется много усилий и времени на адаптацию. |

Таким образом, приложение по автоматизации деятельности школы обладает неплохими показателями качества. Оно демонстрирует хорошие результаты по таким критериям, как функциональность, удобство применения, сопровождаемость и рациональность, что свидетельствует о его способности выполнять основные задачи, удовлетворять потребности пользователей и легко адаптироваться к изменяющимся условиям.

При этом оценка "3" за переносимость указывает на ограниченные возможности использования приложения в других средах или на сложность адаптации к новым условиям функционирования. Это может потребовать дополнительных усилий для интеграции или переноса данных. В целом приложение эффективно выполняет свои функции и подходит для использования в образовательной сфере, однако для повышения его универсальности требуется доработка некоторых аспектов.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы были выявлены и проанализированы требования к разрабатываемому приложению, что позволило заложить надежную основу для создания программного проекта. В рамках этой работы была сформирована ментальная карта (mindmap), основанная на собранных требованиях, а также четко определены задачи, которые были подразделены на спринты и приоритезированы в команде.

Мы получили практический опыт работы в команде с использованием репозитория GitHub, что значительно упростило процесс проектирования структуры приложения и взаимодействия между участниками проекта. Также мы освоили ключевые аспекты тестирования приложений и оценили качество готового продукта, что позволило выявить его сильные и слабые стороны.

Цель курсовой работы, заключающаяся в создании удобного инструмента для обеспечения взаимодействия между всеми участниками образовательного процесса, успешно достигнута. Разработанное приложение включает функции расписания, электронного журнала и электронного дневника, которые создают условия для дальнейшего развития и значительного повышения качества образовательного процесса.

Результаты проведенных лабораторных работ подтвердили, что автоматизация деятельности школы существенно оптимизирует внутренние процессы и повышает общую эффективность управления образовательным учреждением. Разработанная система автоматизации способствует улучшению организации учебного процесса, обеспечивая не только доступ к данным в режиме реального времени, но и упрощая контроль за ключевыми аспектами деятельности школы.

Важным направлением будущей работы над проектом станет реализация улучшенной функциональности приложения. В настоящее время существующая версия продукта требует дальнейших доработок и активного взаимодействия с пользователями для оценки его актуальности и удобства в реальных условиях. Это позволит не только повысить удовлетворенность пользователей, но и внести необходимые коррективы для обеспечения максимальной эффективности в процессе автоматизации школьной деятельности.

СПИСОК ЛИТЕРАТУРЫ

1. Diagram Software and Flowchart Maker [Сайт]. (2022). URL: <https://www.diagrams.net/> (дата обращения: 13.12.2024).
2. Scrum-спринты: все, что вам нужно знать. [Сайт] (2024). URL: https://www.atlassian.com/ru/agile/scrum/sprints (дата обращения: 13.12.2024).
3. Святослав Куликов. Тестирование программного обеспечения. Базовый курс. 3-е издание. (2024).
4. С чего начинается тестирование: что такое тест‑кейс, зачем он нужен и как его писать [Сайт]. (2023). URL: https://practicum.yandex.ru/blog/chto-takoe-test-keys-i-kak-ego-sostavit/#vidy (дата обращения: 13.12.2024).