МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

РАЗРАБОТКА МНОГОПОЛЬЗОВАТЕЛЬСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ГОЛОСОВАНИЯ ЗА ИСХОД СПОРТИВНЫХ СОБЫТИЙ

Курсовое проектирование

по дисциплине «Проектирование автоматизированных систем»

КР.1-53 01 02.10028412.15.81-01

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Самусев Данила Андреевич

(подпись) АСОИ-181

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Крутолевич Сергей Константинович

(подпись)

Дата допуска к защите \_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_

Могилев 2021

Содержание

[Введение 3](#_Toc90411070)

[1 Анализ бизнес процессов 4](#_Toc90411071)

[1.1 Обоснование начала разработки АСОИ 4](#_Toc90411072)

[1.2 Функциональные требования к АСОИ 4](#_Toc90411073)

[1.3 Прочие требования к АСОИ 6](#_Toc90411074)

[2 Проектирование структуры базы данных 8](#_Toc90411075)

[3 Проектирование архитектуры проекта 10](#_Toc90411076)

[3.1 Разработка диаграммы взаимодействия 10](#_Toc90411077)

[3.2 Структура классов АСОИ 10](#_Toc90411078)

[3.3 Диаграмма состояний 12](#_Toc90411079)

[4 Управление процессом разработки программного обеспечения 14](#_Toc90411080)

[4.1 Определение трудоёмкости разработки 14](#_Toc90411081)

[4.2 Отчёт о разработке программных компонентов 15](#_Toc90411082)

[5 Тестирование разработанного ПО 21](#_Toc90411083)

[6 Руководство пользователя 24](#_Toc90411084)

[Заключение 31](#_Toc90411085)

[Список использованных источников 32](#_Toc90411086)

Введение

Целью данной работы является разработка АСОИ. Систему предполагается использовать на рабочих местах членов команды разработки для повышения производительности и скорости разработки. Приложение имеет несколько преимуществ по сравнению со своими аналогами: простота в освоении, гибкость, масштабируемость. Данное приложение может быть использовано небольшими командами разработчиков или, например, организаторами практически любого мероприятия. Пояснительная записка к курсовому проектированию содержит 6 разделов:

* Анализ бизнес процессов – приводится структура формируемых документов в виде таблиц.
* Проектирование структуры базы данных – приводится структура БД в виде таблиц.
* Проектирование архитектуры проекта – приводится структура интерфейсов, запросов и процедур в виде таблиц.
* Управление процессом разработки программного обеспечения – таблицы трудоёмкости и календарный график.
* Тестирование разработанного ПО – раздел включает в себя информацию по тестированию ПО.
* Руководство пользователя – приводится описание процесса формирования документов с формами.
* Заключение – содержит обобщение выполненной работы.

1 Анализ бизнес процессов

* 1. Обоснование начала разработки АСОИ

В качестве объекта автоматизации выбран процесс разработки программного обеспечения.

Система создаётся на основе следующих документов:

– список задач для реализации;

– список ошибок для исправления;

– список реализованных задач для тестирования;

При появлении новых задач руководитель команды разработки добавляет их в систему и назначает на каждую задачу ответственного разработчика. В процессе разработки члены команды продвигают свои задачи через этапы. Руководитель команды отслеживает скорость выполнения задач и производительность команды.

Разработка АСОИ является актуальной, так как позволяет в режиме реального времени отслеживать процесс выполнения задач. Это позволяет:

* правильно распределять задачи между членами команды;
* правильно оценивать скорость разработки;
* своевременно реагировать на возникшие трудности в процессе разработки.

1.2 Функциональные требования к АСОИ

Назначением проекта является проектирование многопользовательской системы. Систему предполагается использовать на рабочих местах членов команды разработки для повышения производительности и скорости разработки. Для управления тестированием могут быть интегрированы различные расширения, позволяющие совместить процессы тестирования и разработки программного обеспечения. Команды могут разбить тестирование и разработку на этапы и повысить его эффективность. Команды контроля качества используют задачи, настраиваемые экраны, поля и процессы для управления ручным и автоматическим тестированием.

Систему предполагается создать с целью упростить управление рабочим процессом для самых различных команд, для улучшения качества обслуживания заказчика.

Критерий оценки достижений целей системы – увеличение числа выполненных задач. Так как система позволит увеличить скорость разработки, то увеличится количество реализованного функционала.

Система должна удовлетворять следующим требованиям:

– Надежности;

– Безопасности;

– Требования к защите информации от несанкционированного доступа. Должна быть предусмотрена защита от несанкционированного доступа к данным, ввода данных, их удаления;

– Система должна быть доступна с любого компьютера в сети компании или команды разработки;

– Информация, хранящаяся в системе, должна быть защищена от аварийных ситуаций, влияния внешних воздействий (радиоэлектронная защита);

– Информация, хранящаяся в системе, должна быть защищена от аварийных ситуаций, влияния внешних воздействий (радиоэлектронная защита);

– Квалификация персонала. Персонал должен быть обучен правилам работы с системой, при этом не обязательно наличие специального технического образования.

Функции, выполняемые подсистемами объектов автоматизации:

– Отображение задач в различных статусах. Без использования автоматизированной системы поиск активных приоритетных задач занимает значительное время. И для хранения самих формуляров необходимы дополнительные площади. Автоматизированная система хранит эту информацию сколь угодно долго, поиск осуществляется за считанные секунды, не требуется дополнительная площадь.

– Создание новых задач, контроль их статуса. При использовании автоматизированной системы контроль над текущими задачами будет осуществляться легко и быстро.

– Проверка задач, закрепленных за работником. Автоматизированная система хранит информацию о задачах сколь угодно долго, поиск и проверка будет осуществляться за считанные секунды. Работник также сможет видеть список задач, закрепленных за ним.

– Перевод задач в различные статусы позволяет каждому члену команды разработки концентрироваться только на ему нужных задачах.

В качестве функциональных требований выступают требования формирования страницы «Описание задачи».

Таблица 1.1 — Поля страницы «Описание задачи»

|  |  |
| --- | --- |
| Поле на странице | Обозначение |
| Название | [Name] |
| Состояние | [State] |
| Владелец | [Owner] |
| Описание | [Description] |
| Комментарии | [Comments] |
| Категория | [Category] |

Таблица 1.2 — Поля страницы «Список задач»

|  |  |
| --- | --- |
| Поле на странице | Обозначение |
| Имя | [Name] |
| Состояние | [State] |
| Категория | [Category] |

Таблица 1.3 — Поля страницы «Список членов команды»

|  |  |
| --- | --- |
| Поле на странице | Обозначение |
| Имя | [Name] |
| Фамилия | [Surname] |
| Адрес электронной почты | [Email] |
| Роль | [Role] |

1.3 Прочие требования к АСОИ

Система должна быть адаптивной к изменениям и простой в использовании, для большего ее распространения.

Минимальные системные требования:

1. Процессор: двухъядерный Intel с тактовой частотой 2,2 ГГц или двухъядерный AMD с тактовой частотой 2,5 ГГц

2. Оперативная память: 1,5 ГБ (для XP), 2 ГБ (для Vista/7)

3. Видеокарта: GeForce 8800 (с 512 МБ видеопамяти) или Radeon HD3850 (с 512 МБ видеопамяти)

4. Жесткий диск: 1.2 ГБ свободного места

Рекомендуемые системные требования:

1. Процессор: Intel Core i7 9700KF 3.6GHz

2. Оперативная память: 16 ГБ (для Windows 10)

3. Видеокарта: GeForce RTX 3080 (с 8 ГБ видеопамяти)

4. Жесткий диск: 1.2 ГБ свободного места

2 Проектирование структуры базы данных

С использованием всей предоставленной документации была разработана структура базы данных. Все таблицы были приведены к третьей нормальной форме.

Для каждой таблицы, которая есть в базе данных, ниже приведена структура.

Таблица 2.1 – Структура таблицы “Пользователи”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK | Name | Type | NotNull | Unique | Lenght |
| + | Id | int | + | + |  |
|  | Name | varchar | + |  | 150 |
|  | Password | varchar | + |  | 150 |
|  | Surname | varchar | + |  | 150 |
|  | Email | varchar | + |  | 150 |

Таблица 2.2 – Структура таблицы “Задачи”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK | Name | Type | NotNull | Unique | Lenght |
| + | Id | int | + | + |  |
|  | Name | varchar | + |  | 150 |
|  | Description | varchar | + |  | 150 |
|  | UrgencyId | int | + |  |  |
|  | CategoryId | int | + |  |  |
|  | UserId | int | + |  |  |
|  | StateId | int | + |  |  |
|  | ProjectId | int | + |  |  |
|  | SprintId | int | + |  |  |

Таблица 2.3 – Структура таблицы “Проекты”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK | Name | Type | NotNull | Unique | Lenght |
| + | Id | int | + | + |  |
|  | Name | varchar | + |  | 150 |

Таблица 2.4 – Структура таблицы “Истории”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK | Name | Type | NotNull | Unique | Lenght |
| + | Id | int | + | + |  |
|  | Action | varchar | + |  | 50 |
|  | TaskId | int | + |  |  |
|  | UserId | int | + |  |  |

Таблица 2.5 – Структура таблицы “Комментарии”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PK | Name | Type | NotNull | Unique | Lenght |
| + | Id | int | + | + |  |
|  | Text | varchar | + |  | 250 |
|  | TaskId | int | + |  |  |
|  | UserId | int | + |  |  |

Для отображения информационной модели рассматриваемого процесса используются следующие сущности:

- «Пользователи». Данная сущность хранит в себе всю информацию, которая относится к пользователям, относящимся к конкретному проекту.

- «Задачи». Данная сущность содержит в себе всю информацию созданных пользователями задачах.

- «Проекты». Данная сущность содержит информацию проектах, с которыми работают пользователи. Именно для проектов создаются задачи.

- «Истории». Данная сущность содержит информацию об истории изменения задач (изменении статуса, имени и т.д.).

- «Комментарии». Данная сущность представляет собой комментарий, который может быть оставлен к какой-либо задаче, например после её выполнения.

3 Проектирование архитектуры проекта

3.1 Разработка диаграммы взаимодействия

Для описания процессов, происходящих в web-приложении, была разработана диаграмма взаимодействия. Она описывает все реализованные на данный момент варианты использования приложения, которые доступны пользователю. Диаграмма взаимодействия представлена в графической части.

В таблице 3.1 перечислены классы, реализованные в ходе разработки приложения.

Таблица 3.1 — Классы АСОИ

|  |  |
| --- | --- |
| Имя класса | Стереотип класса |
| Role | boundary |
| Task | boundary |
| DetailedTicket | boundary |
| Project | boundary |
| User | boundary |
| Colaborator | boundary |

3.2 Структура классов АСОИ

Диаграмма классов – это диаграмма, которая демонстрирует общую структуру классов, их атрибутов и взаимосвязей между ними.

Атрибуты класса определяют состав и структуру данных, которые хранятся в объектах этого класса. Каждый атрибут имеет имя и тип, определяющий, какие данные он представляет.

Таблица 3.2 — Интерфейс класса UserController

|  |  |
| --- | --- |
| Класс UserController | |
| Поля | |
| Имя | Тип данных |
| \_usersToProjectsService | IUsersToProjectsService |
| \_userService | IUserService |
| Методы | |
| Имя | Тип данных |
| GetTeamData | Task<IActionResult> |
| GetAllColaborators | Task<IActionResult> |
| AddColaborator | Task<IActionResult> |
| DeleteColaborator | Task<IActionResult> |

Таблица 3.3 — Интерфейс класса TicketController

|  |  |
| --- | --- |
| Класс TicketController | |
| Поля | |
| Имя | Тип данных |
| \_ticketService | ITicketService |
| Методы | |
| Имя | Тип данных |
| GetDetailedTicket | Task<IActionResult> |
| CreateTicket | Task<IActionResult> |
| GetTicketsPreviews | Task<IActionResult> |
| UpdateTicket | Task<IActionResult> |
| DeleteTicket | Task<IActionResult> |

Таблица 3.4 — Интерфейс класса ProjectController

|  |  |
| --- | --- |
| Класс ProjectController | |
| Поля | |
| Имя | Тип данных |
| \_projectService | IProjectService |
| Методы | |
| Имя | Тип данных |
| GetProject | Task<IActionResult> |
| GetAllUserOwnProjects | Task<IActionResult> |
| GetAllUserColaborate | Task<IActionResult> |
| UpdateProject | Task<IActionResult> |
| CreateProject | Task<IActionResult> |
| DeleteProject | Task<IActionResult> |

Таблица 3.5 — Интерфейс класса AccountController

|  |  |
| --- | --- |
| Класс AccountController | |
| Поля | |
| Имя | Тип данных |
| \_userService | IUserService |
| Методы | |
| Имя | Тип данных |
| SignUp | Task<IActionResult> |
| SignIn | Task<IActionResult> |

Перечисленные выше классы, которые представляют собой контроллеры, содержат всю основную бизнес логику данного приложения. Они позволяют просматривать, изменять, добавлять и удалять информацию.

3.3 Диаграмма состояний

Диаграмма состояний используется для описания взаимодействия между программой и пользователем. Диаграмма моделирует переходы между диалоговыми формами и список внутренних действий в форме. На переходах между формами отображается имя кнопки, вызвавшей событие перехода.

Основными элементами диаграммы состояний являются «Состояние» и «Переход». Диаграмма состояний похожа на диаграмму деятельности, но деятельность в случае диаграммы состояний заменена состоянием, переходы символизируют действия.

Состояние содержит имя или имя и список внутренних действий. Список внутренних действий содержит перечень действий или деятельностей, которые выполняются во время нахождения объекта в данном состоянии. Данный список фиксированный.

Список основных действий включает в себя:

* Enter web-site – действие, которое выполняется при открытии веб-сайта;
* Open page – действие, которое выполняется при открытии страницы веб-сайта;
* Exit page – действие, которое выполняется при закрытии страницы веб-сайта.

4 Управление процессом разработки программного обеспечения

4.1 Определение трудоёмкости разработки

В таблице 4.1 указаны примерные трудозатраты разработки элементов АСОИ: количество часов, потраченное на создание того или иного элемента АСОИ. В таблице 4.2 указан календарный план разработки элементов АСОИ.

Таблица 4.1 — Трудоёмкость разработки программного обеспечения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Компонент | Число полей | Число диалоговых элементов | Число вычисляемых процедур | Число страниц печатной формы | Трудоёмкость разработки, ч |
| «table» tCategories | 2 |  |  |  | 0,3 |
| «table» tComments | 4 |  |  |  | 0,3 |
| «table» tFeedbacks | 5 |  |  |  | 0,4 |
| «table» tHistories | 4 |  |  |  | 0,3 |
| «table» tProjects | 2 |  |  |  | 0,3 |
| «table» tRoles | 2 |  |  |  | 0,3 |
| «table» tSprints | 5 |  |  |  | 0,5 |
| «table» tStates | 2 |  |  |  | 0,2 |
| «table» tTasks | 9 |  |  |  | 0,6 |
| «table» tUrgencies | 2 |  |  |  | 0,2 |
| «table» tUsers | 5 |  |  |  | 0,5 |
| fLogin |  | 6 |  |  | 5 |
| fProject |  | 3 |  |  | 3 |
| fTicket |  | 9 |  |  | 10 |
| docUsers | 6 |  |  | - | 11 |
| docTickets | 5 |  |  |  | 12 |
| Итого | | | | | 45 |

Таблица 4.2 — Календарный план разработки

|  |  |  |
| --- | --- | --- |
| Наименование компонента | Срок выполнения | |
|  | Начало г. | Окончание г. |
| «table» tCategories | 03.10.2020 | 04.10.2020 |
| «table» tComments | 03.10.2020 | 04.10.2020 |
| «table» tFeedbacks | 03.10.2020 | 04.10.2020 |
| «table» tHistories | 03.10.2020 | 04.10.2020 |
| «table» tProjects | 03.10.2020 | 04.10.2020 |
| «table» tRoles | 03.10.2020 | 04.10.2020 |
| «table» tSprints | 03.10.2020 | 04.10.2020 |
| «table» tStates | 03.10.2020 | 04.10.2020 |
| «table» tTasks | 03.10.2020 | 04.10.2020 |
| «table» tUrgencies | 03.10.2020 | 04.10.2020 |
| «table» tUsers | 03.10.2020 | 04.10.2020 |
| fLogin | 06.10.2020 | 8.10.2020 |
| fProject | 09.10.2020 | 10.10.2020 |
| fTicket | 11.10.2020 | 15.10.2020 |
| docUsers | 16.10.2020 | 20.10.2020 |
| docTickets | 21.10.2020 | 25.10.2020 |

4.2 Отчёт о разработке программных компонентов

В разработке использовался обширный стек технологий. В него входят:

* платформа .NET 5.0 и её основной язык C#
* Web-фреймворк ASP.NET
* ORM Dapper
* фреймворк для клиентской части приложения React JS
* Язык запросов SQL

Несколько примеров контроллеров. Эти классы служат для приёма и обработки WEB запросов.

Контроллер авторизации AccountController

namespace JustInMindApp.Controllers

{

[Route("[controller]")]

[Produces("application/json")]

[ApiController]

public class AccountController : ControllerBase

{

private readonly IUserService \_userService;

public AccountController(IUserService userService)

{

\_userService = userService;

}

[HttpPost("signUp")]

public async Task<IActionResult> SignUp([FromBody] SignUpRequest request)

{

var user = await \_userService.GetByEmailAsync(request.Email);

if (user != null)

{

return BadRequest("User with entered email already exists!");

}

var newUser = new User

{

Name = request.Name,

Surname = request.Surname,

Email = request.Email,

Password = request.Password,

};

await \_userService.InsertAsync(newUser);

return Ok();

}

[HttpPost("signIn")]

public async Task<IActionResult> SignIn([FromBody] SignInRequest request)

{

var user = await \_userService.GetByEmailAndPasswordAsync(request.Email, request.Password);

if (user == null)

{

return NotFound("User is not found!");

}

var token = TokenCreater.CreateToken(user);

var encodedToken = new JwtSecurityTokenHandler().WriteToken(token);

var response = new

{

token = encodedToken,

userName = token.Claims.FirstOrDefault(c => c.Type == ClaimsIdentity.DefaultNameClaimType).Value,

userId = token.Claims.FirstOrDefault(c => c.Type == nameof(JustInMind.Shared.Models.User.Id).ToLower()).Value,

};

return new ObjectResult(response);

}

}

}

Пример класса-сервиса. Он выступает прослойкой между WEB-частью и частью доступа к базе данных.

public class ProjectService : IProjectService

{

private readonly IProjectRepository \_projectRepository;

public ProjectService(IProjectRepository projectRepository)

{

\_projectRepository = projectRepository;

}

public async Task<Project> GetAsync(int id)

{

return await \_projectRepository.GetAsync(id);

}

public async Task<IEnumerable<Project>> GetAllAsync(int userId)

{

return await \_projectRepository.GetAllAsync(userId);

}

public async System.Threading.Tasks.Task InsertAsync(CreateProjectRequest request)

{

await \_projectRepository.InsertAsync(request);

}

public async Task<bool> UpdateAsync(Project entity)

{

return await \_projectRepository.UpdateAsync(entity);

}

public async Task<bool> DeleteAsync(Project entity)

{

return await \_projectRepository.DeleteAsync(entity);

}

public async Task<bool> LeaveAsync(UsersToProjects entity)

{

return await \_projectRepository.LeaveAsync(entity);

}

}

Пример репозитория. Репозитории выступают классами, которые предоставляют доступ к данным в хранилище. В нашем случае это MS SQl Server база.

public class ProjectRepository : IProjectRepository

{

private readonly string connectionString;

public ProjectRepository(string connectionString)

{

this.connectionString = connectionString;

}

public async Task<Project> GetAsync(int id)

{

using var db = new SqlConnection(connectionString);

var project = await db.GetAsync<Project>(id);

return project;

}

public async Task<IEnumerable<Project>> GetAllAsync(int userId)

{

string sql = "SELECT p.Id, p.Name, up.UserId, up.UserRoleId FROM UsersToProjects up " +

"LEFT JOIN Projects p ON p.Id = up.ProjectId " +

"WHERE up.UserId = @UserId";

using var db = new SqlConnection(connectionString);

var projects = await db.QueryAsync<Project>(sql, new { UserId = userId });

return projects;

}

public async Task<int> InsertAsync(CreateProjectRequest request)

{

using var db = new SqlConnection(connectionString);

string sql = "INSERT INTO Projects (Name) " +

"OUTPUT INSERTED.Id " +

$"VALUES ('{request.Name}')";

var projectId = await db.QuerySingleAsync<int>(sql);

sql = "INSERT INTO UsersToProjects (ProjectId, UserId, UserRoleId) " +

"OUTPUT INSERTED.Id " +

$"VALUES ({projectId}, {request.OwnerId}, 5)";

return await db.QuerySingleAsync<int>(sql);

}

public async Task<bool> UpdateAsync(Project entity)

{

using var db = new SqlConnection(connectionString);

return await db.UpdateAsync(entity);

}

public async Task<bool> DeleteAsync(Project entity)

{

using var db = new SqlConnection(connectionString);

return await db.DeleteAsync(entity);

}

public async Task<bool> LeaveAsync(UsersToProjects entity)

{

using var db = new SqlConnection(connectionString);

return await db.DeleteAsync(entity);

}

}

Пример модели сущности в базе данных.

public class Project

{

[Key]

public int Id { get; set; }

public string Name { get; set; }

public int UserId { get; set; }

public int UserRoleId { get; set; }

}

5 Тестирование разработанного ПО

Для нормальной работы системы необходимо 512 Мбайт оперативной памяти, Windows 7 и выше, MS SQLServer 2018 и выше, браузер Google Chrome (либо любой современный браузер) актуальной версии, фреймворк .Net 5.

Результатом тестирования в данном случае может послужить безошибочное добавление нового пользователя, задачи и т.д.

После тестирования получили результаты:

* Информационная система поддерживает многопользовательский режим.
* Тестирование запросов прошло успешно.

В курсовом проекте разработана база данных с клиентскими приложениями, которая хранится на SQL сервере. Составлены запросы и формы. Формы составлены на основе запросов и таблиц, и используются для занесения и модификации информации в базе данных.

Таблица 5.1 — Результаты тестирования АСОИ

|  |  |  |
| --- | --- | --- |
| №  п/п | Описание теста | Ожидаемый результат |
| 1. Модуль Авторизации | | |
| 1.1 | В адресной строке браузера Google Chrome версии не ниже 52.0.2743.хх . ввести адрес сайта. https://justinmind.azurewebsites.net/ | Отображается окно авторизации. |
| 1.2 | В форме поля почта пользователя и пароль пустые. Нажать кнопку «Войти». | Появилось предупреждение Text  Description automatically generated with medium confidence  Загрузка главной страницы не произошла |
| 1.3 | В форме авторизации ввести неверные почту пароль.  Нажать кнопку «Войти». | Появилось предупреждение  “Пользователь не найден” |
| 1.4 | В форме авторизации почту пользователя и пароль.  Нажать кнопку «Войти». | Загрузится Главная страница приложения. |
| 1.5 | Нажать кнопку | Загрузилась страница регистрации |
| 1.6 | Одно из полей формы регистрации пустое.  Нажать кнопку “Зарегистрироваться” | Появилось предупреждение  “Не все обязательные поля заполнены” |
| 1.7 | В поле “почта” введёно существующее значение.  Нажать кнопку ‘Регистрация” | Появляется предупреждение  A picture containing graphical user interface  Description automatically generated |
| 1.8 | Все поля а форме регистрации заполнены корректно | Появляется сообщение A picture containing graphical user interface  Description automatically generated |
| 2 Тестирование страницы с задачами | | |
| 2.1 | Нажать кнопку Text  Description automatically generatedна панели инструментов | Появляется всплывающее окно-форма создания нового проекта |
| 2.2 | Нажать на кнопку Text  Description automatically generatedна панели инструментов | Появляется список доступных проектов |
| 2.3 | Нажать на один из вариантов при выборе проекта | Происходит обновление страницы, появляются новые инструменты. Отображаются колонки с задачами |
| 2.4 | Нажать на кнопку Text  Description automatically generated with medium confidenceна панели инструментов | Появляется окно подтверждения для удаления |
| 2.5 | Нажать на кнопку добавления задачи | Тут же кнопка меняется на мини-форму с полем для имени задачи |
| 2.6 | В мини-форме нажать кнопку “добавить задачу” | Новая задача появляется на доске |
| 2.7 | Перетянуть задачу из одной колонки в другую | Задача появляется в новой колонке |
| 2.8 | Нажать на иконку мусорки на карточке задачи | Задача удаляется из доски навсегда |
| 2.9 | Нажать на кнопку карандаша рядом с задачей | Открывается всплывающая форма редактирования задачи |
| 2.10 | Нажать на кнопку | Появляется форма создания нового спринта |
| 3 Тестирование вспомогательных форм | | |
| 3.1 | В форме добавления проекта ввести имя(доступно пустое имя) | Новый проект появляется в списке проектов |
| 3.2 | В форме подтверждения удаления проекта нажать кнопку disagree | Проект не удаляется |
| 3.3 | В форме подтверждения удаления проекта нажать кнопку agree | Проект удаляется из базы и из списка всех проектов |
| 3.4 | В выпадающем списке проектов нажать на 1-ый проект | Приложение воспринимает выбор нового проекта и перерисовывается |
| 3.5 | В форме добавления спринта одно из полей ввода пустое | Появляется ошибка |
| 3.6 | В форме добавления спринта все поля заполнены корректно | Новый спринт добавляется в конец списка спринтов |
| 3.7 | В форме редактирования задачи изменить имя. Нажать кнопку сохранить | Изменения видны на доске задач |
| 3.8 | На форме редактирования задачи нажать кнопку крестика в верхнем углу | Форма закрывается без изменений |

6 Руководство пользователя

Работа в приложении начинается с логина в приложении, если вы ещё не залогинились. В этом случае приложение вас предупредит об этом и переадресует на страницу Login

Graphical user interface, text, application, Word

Description automatically generated

Рисунок 1 – Окно предупреждения о необходимости входа

Graphical user interface, application

Description automatically generated

Рисунок 2 – Окно для логина

При вводе некорректных данных появится всплывающее окошко. Подобного рода сообщения появляются на любое некорректное действие пользователя.

Graphical user interface, application

Description automatically generated

Рисунок 3 – Окно неправильного ввода данных при логине

При вводе корректных данных происходит переадресация на основную часть приложения. На ней представлено меню управления проектом, текущие задачи и небольшой личный кабинет пользователя. При клике на кнопку личного кабинета всплывает окно, позволяющее разлогиниться. Меню предоставляет доступ к проектам и ко всем членам команды. Эта таблица предоставляет богатый функционал: удаление одного/множества пользователей, приглашение нового члена команды, удобный механизм сортировки и пагинации.

Graphical user interface, application

Description automatically generated

Рисунок 18 – Основное окно приложения

Graphical user interface, application, Teams

Description automatically generated

Рисунок 19 – Меню

A picture containing shape

Description automatically generated

Рисунок 20 – Кнопка личного кабинета

Graphical user interface, application

Description automatically generated

Рисунок 21 – Личный кабинет

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 22 – Задачи проекта

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 23 – Список всех сотрудников

Задачи можно перетаскивать в различные состояния. Например, перетащим первую задачу из первого состояния в другое: видно, что задача приобрела новое состояние. Graphical user interface

Description automatically generated

Рисунок 24 – Новое состояние задачи “Investigation”

Так же каждый тип задачи имеет свой определённый цвет. При необходимости можно добавить новую задачу нажав на кнопку Add Task. Так же задачи можно удалять (иконка мусорного ведра) и редактировать (иконка карандаша).

Graphical user interface, application

Description automatically generated

Рисунок 25 – Добавление новой задачи

Задачи можно осмотреть подробнее при нажатии на иконку карандаша, после этого появится всплывающее меню с описанием задачи. Там будут представлены описание задачи, её имя, владелец, комментарии, тип и её статус. Все эти данные могут быть изменены.

Graphical user interface

Description automatically generated

Рисунок 26 – Подробное описание задачи

Для того, чтобы пригласить пользователя в качестве коллаборатора, нужно нажать на соответствующую кнопку на таблице с пользователями. Появится специальная формочка, туда нужно ввести электронную почту пользователя.

Graphical user interface, application

Description automatically generated

Рисунок 27 – Форма приглашения пользователя

Заключение

В ходе разработки курсового проекта был создан программный модуль, необходимый для оптимизации и ускорения процесса разработки некоторого продукта. Написанное приложение имеет некоторые преимущества, по сравнению с похожими инструментами, например: простота в освоение, подходит для небольших команд, легковесность, возможность удалённого доступа.

Список использованных источников

* 1. Маркин, А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин. - М.: Диалог-Мифи, 2014. - 384 c.
  2. [Бхамидипати](http://oz.by/books/more106510.html?id_search=127378#writer), К. SQL. Справочник программиста/ К.Бхамидипати–М.: Эком, 2003. – 304 с.
  3. Бьюли, А. Изучаем SQL / А.Бьюли, Э. Оппель. – М.: Символ, 2007. – 312 с.
  4. Жилинский, А. Самоучитель Microsoft SQL Server 2005 / А.Жилинский – СПб: BHV, 2004. – 224 с.
  5. Павловская, Т. А. С#. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2007. – 432 с.: ил.
  6. Подбельский, В. В. Язык С#. Решение задач / В. В. Подбельский. – М.: Инфра-М, 2014. – 296 с.
  7. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, **2017**. - 480 c.
  8. Еремин И.И., Астафьев Н.Н. Введение в теорию линейного и выпуклого программирования -М.: Наука, 1976 г. – 239 с.
  9. Общий курс высшей математики для экономистов. Учебник / под ред В.И. Ермакова.- М.: ИНФА - М. - 656 с. - (серия «высшее образование»).

10. [https://arman-engineering.ru/baza-znanij-inzhenera-avtomatizirovannaja-sistema-upravlenija-tehnologicheskimi-processami]