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:

1. – Надежности;
2. – Безопасности;
3. – Требования к защите информации от несанкционированного доступа. Должна быть предусмотрена защита от несанкционированного доступа к данным, ввода данных, их удаления;
4. – Система должна быть доступна с любого компьютера в сети компании или команды разработки;
   1. – Информация, хранящаяся в системе, должна быть защищена от аварийных ситуаций, влияния внешних воздействий (радиоэлектронная защита);
   2. – Информация, хранящаяся в системе, должна быть защищена от аварийных ситуаций, влияния внешних воздействий (радиоэлектронная защита);
5. – Квалификация персонала. Персонал должен быть обучен правилам работы с системой, при этом не обязательно наличие специального технического образования.
6. Функции, выполняемые подсистемами объектов автоматизации:
7. – 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.
8. – Создание новых задач, контроль их статуса. При использовании автоматизированной системы контроль над текущими задачами будет осуществляться легко и быстро.
9. – 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.
10. – 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

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.

Функциональные требования:

1. Отображение задач в различных статусах.
2. Создание новых задач, контроль их статуса.
3. Проверка задач, закрепленных за работником.
4. Перевод задач в различные статусы позволяет каждому члену команды разработки концентрироваться только на ему нужных задачах.
5. Регистрация новых пользователей.
6. Удаление пользователей.
7. Удаление проектов.
8. Создание проектов.
9. Изменение данных проекта.