**СОДЕРЖАНИЕ**

[Перечень условных обозначений 5](#_Toc164800851)

[Введение 6](#_Toc164800852)

[1 Информационные системы учета и отслеживания дефектов программных продуктов 7](#_Toc164800853)

[1.1 Системы учета и отслеживания дефектов программных продуктов в процессе разработки программного обеспечения. Анализ предметной области 7](#_Toc164800854)

[1.2 Существующие аналоги системы отслеживания дефектов программных продуктов 10](#_Toc164800855)

[1.3 Выводы и постановка задач на дипломное проектирование 18](#_Toc164800856)

[2 Эргономическое проектирование информационной системы 20](#_Toc164800857)

[2.1 Разработка алгоритмов работы пользователя 20](#_Toc164800858)

[2.2 Разработка эргономических требований и сценария информационного взаимодействия 31](#_Toc164800859)

[2.3 Эргономическая оценка проектируемой системы и выводы 43](#_Toc164800860)

[3 Разработка информационной системы 49](#_Toc164800861)

[3.1 Структура системы 49](#_Toc164800862)

[3.2 Реализация клиентской стороны 52](#_Toc164800863)

[3.3 Структура базы данных информационной системы 56](#_Toc164800864)

[3.4 Выводы и оценка результатов разработки 61](#_Toc164800865)

[4 Тестирование информационной системы 62](#_Toc164800866)

[5 Технико-экономическое обоснование эффективности разработки и использования информационной системы отслеживания дефектов программных продуктов 80](#_Toc164800867)

[5.1 Характеристика информационной системы 80](#_Toc164800868)

[5.2 Расчет сметы затрат на разработку информационной системы 81](#_Toc164800869)

[5.3 Расчет экономического эффекта от использования информационной системы 83](#_Toc164800870)

[6 Охрана труда. Обоснование выбора системы оповещения о пожаре в производственном помещении 87](#_Toc164800871)

[Заключение 91](#_Toc164800872)

[Список использованных источников 92](#_Toc164800873)

[Приложение А (обязательное) Листинг программы 92](#_Toc164800873)

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ПО – программное обеспечение

ПС – программное средство

ПП – программный продукт

СОДПП – система отслеживания дефектов программных продуктов

ПК – персональный компьютер

ОС – операционная система

ЭТ – эргономические требования

СЧКС – система «человек-компьютер-среда»

БД – база данных

ГПИ – графический пользовательский интерфейс

ВВЕДЕНИЕ

Оценка качества ПП является неотъемлемой частью цикла разработки ПО. Упущенные или необработанные дефекты могут привести к серьезным временным и материальным потерям, в крайнем случае — судебным разбирательствам, что, губительно для любого производства или бизнеса.

В ходе данного дипломного проекта будет разработана информационная система отслеживания дефектов программных продуктов, с расчетом на дальнейшую публикацию с открытым исходным кодом. Написание фундаментальной структуры базового решения и функциональности на популярном стеке технологий позволит сторонним разработчикам легко вносить изменения, добавлять новые функции и модули, а также оптимизировать работу приложения под конкретные потребности пользователей или бизнес-процессы организации.

Актуальность выбранной темы обуславливается интенсивной автоматизации работ трудно поддающихся формализации и общей автоматизации разработки ПО. В контексте информационных технологий это означает потребность в разработке и усовершенствовании эффективных сервисов, которые направлены на облегчение работы разработки программного обеспечения в целом, и отдела тестирования в частности.

Исходя из динамики прошлых лет видно, что рынок требует не только усовершенствования старых систем, но и разработки новых, более гибких систем. Разрабатываемое ПО несет цель облегчить и систематизировать часть процессов, связанных с отслеживанием и учетом программных дефектов. Это включает в себя частичную автоматизацию таких задач, как отслеживание и управление дефектами программных продуктов, анализ их воздействия, управление приоритетами, мониторинг процессов устранения дефектов, а также оценка эффективности и вовлечение членов команды разработки. Автоматизируя эти процессы, организации могут уменьшить количество ошибок, связанных с управлением дефектами, повысить точность информации, повысить эффективность и, в конечном итоге, улучшить общий опыт участников в разработке программного обеспечения.

Цель разработки данного программного средства заключается в предоставлении легко модифицируемого базового решения в открытый доступ, что в дальнейшем позволит удовлетворить обще-рыночную потребность. Данное ПС позволит структурированно хранить, обрабатывать и изменять данные, необходимые для функционирования отдела тестирования предприятия или другого отдела, схожего по назначению.

1. ИНФОРМАЦИОННЫЕ СИСТЕМЫ УЧЕТА И ОТСЛЕЖИВАНИЯ ДЕФЕКТОВ ПРОГРАММНЫХ ПРОДУКТОВ

1.1 Системы учета и отслеживания дефектов программных продуктов в процессе разработки программного обеспечения. Анализ предметной области

Область разработки ПО постоянно расширяется и вместе с ней растет процент программ, используемых в самом процессе цикла разработки.

На сегодняшний день компании, занимающиеся разработкой программного обеспечения, сталкиваются с растущим объемом кода, частыми обновлениями и необходимостью поддерживать высокий уровень стабильности продукта. Тестирование, исправление дефектов и интеграция нового функционала являются неотъемлемой частью работы любого разработчика и беспрерывно сопровождают продукт на каждой стадии цикла разработки программного обеспечения.

На этапе тестирования современного цикла разработки программного обеспечения могут возникнуть различные проблемы в контексте учета и отслеживания дефектов:

* замедление процесса исправления ошибок, назначения персонала и контроля состояния дефектов, связанные с отсутствием совместной среды учета дефектов;
* затруднение взаимопонимания и рабочих коммуникаций в связи с отсутствием согласованного формата и стандартизированного шаблона содержания отчета;
* необходимость дополнительных временных затрат, связанных с отсутствием приоритизации и детального описания дефектов;

Системы отслеживания дефектов решают все из перечисленных проблем, и являются неотъемлемой частью инфраструктуры разработки программных продуктов, предоставляя возможность эффективного управления дефектами от момента их обнаружения до их полного исправления. Однако, большинство существующих программных решений разработаны с расчетом на использование в крупных компаниях, и не поддерживают возможность автономной работы.

Разрабатываемый программный продукт нацелен на упрощение и систематизацию определенных этапов процесса отслеживания и учета дефектов в программном коде. Это включает частичную автоматизацию таких задач, как мониторинг и управление дефектами программных продуктов, анализ их воздействия, управление приоритетами, отслеживание процессов устранения дефектов, а также оценку эффективности и вовлечение участников команды разработки.

За счет интеграции системы отслеживания дефектов в рабочее окружение разработки, организации способны уменьшить количество ошибок, связанных с управлением дефектами, повысить точность предоставляемой информации, улучшить эффективность, повысить общий уровень автоматизации трудового процесса, а так же облегчить участникам цикла разработки некоторые из повседневных задач.

При тестировании программных продуктов, а так же в процессе внедрения нового функционала, разработкой которого занималось более одного человека, очевидна необходимость коммуникации и общего взаимодействия работников. В разрезе разрабатываемого решения коммуникация не обязательно должна выглядеть как групповое обсуждение дефекта [1], а может представлять собой просто опосредованное общение в виде отчета по дефекту или, в нашем случае, экземпляр учтенного дефекта в СОДПП.

Большинство из существующих приложений доступны через веб-интерфейс, что делает их доступными из любого браузера без необходимости установки дополнительного программного обеспечения, но и делает их менее гибкими в замкнутых коммерческих средах, в которых по каким-либо причинам ограничен доступ к внешнему окружению. Для хранения информации о дефектах используются различные типы баз данных, такие как SQL – подобные или NoSQL, что зависит от потребностей масштабируемости и производительности проекта.

Для реализации вышеперечисленного функционала достаточно одной общей БД вне зависимости от ее класса [2], она может быть как централизованная, так и распределенная. Современным и наилучшим решением будет применение отдельной БД на ПК каждого пользователя, которая может быть в дальнейшем связана между собой — это может быть как посредник в виде общего сервера, так и прямая связь между клиентами пользователей. В существующих решениях чаще всего используется первый вариант, а сервер для обработки запросов от пользователей и управления данными в основном [3] используются следующие языки программирования:

* JavaScript и TypeScript;
* Python;
* C# и платформа .NET;
* Java;
* Ruby (Ruby on Rails);

Тем не менее, приложение обычно разрабатывается с уже известным команде разработчиков набором (далее – стеком) технологий, или максимально похожим на него. Это происходит в силу принятой индустриальной практики, при которой команды разработчиков стремятся к максимальной эффективности и оптимизации процесса создания программного обеспечения.

Если стек технологий достаточно гибок для реализации требуемого функционала и способен беспроблемно интегрироваться в требуемое окружение, то предпочтение отдается знакомым инструментам, что позволяет избежать излишних затрат времени и ресурсов на изучение новых стеков для каждого нового проекта. Такой подход содействует ускорению процесса разработки, минимизации рисков и обеспечивает более высокое качество конечного продукта благодаря глубокому знанию выбранного стека технологий со стороны команды разработчиков. Анализируемая сфера разработки программного обеспечения так же подчинена такому подходу.

Поскольку данные о дефектах могут содержать чувствительную информацию о продукте или организации, некоторые из крупных компаний закупают решения для сугубо внутреннего пользования, что существенно затрудняет доступ к некоторой статистике.

Разрабатываемый программный продукт предоставит базовое решения, что в дальнейшем позволит удовлетворить обще-рыночную потребность. Данный ПП позволит структурированно хранить, обрабатывать и в режиме реального времени изменять данные, необходимые для функционирования отдела тестирования предприятия или другого отдела, схожего по назначению. Преимуществами системы являются ее легкая модифицируемость и масштабируемость.

1.2 Существующие аналоги системы отслеживания дефектов программных продуктов

В области отслеживание дефектов ПП наблюдается некоторая стагнация — большинство представленных в свободном доступе вариантов были разработаны более десяти лет назад. Список популярных существующих программных решений ограничивается списком из четырех названий:

1. Bugzilla
2. JIRA
3. Trac

4. The Bug Genie

Далее рассмотрим их более детально, вкратце описав и выделив ключевые характеристики в каждом из них:

1. Bugzilla – это одна из наиболее знаменитых систем отслеживания ошибок и дефектов в программном обеспечении. В 1998 году Netscape представила первую версию этой системы в общий доступ. Bugzilla является свободным ПО и распространяется по публичной лицензии, то есть пользователи имеют права на его неограниченную установку, запуск, свободное использование [4], но не могут просматривать его исходный код, что значительно усложняет его модернизацию. Разработку этой системы сейчас ведет Mozilla Foundation [5].

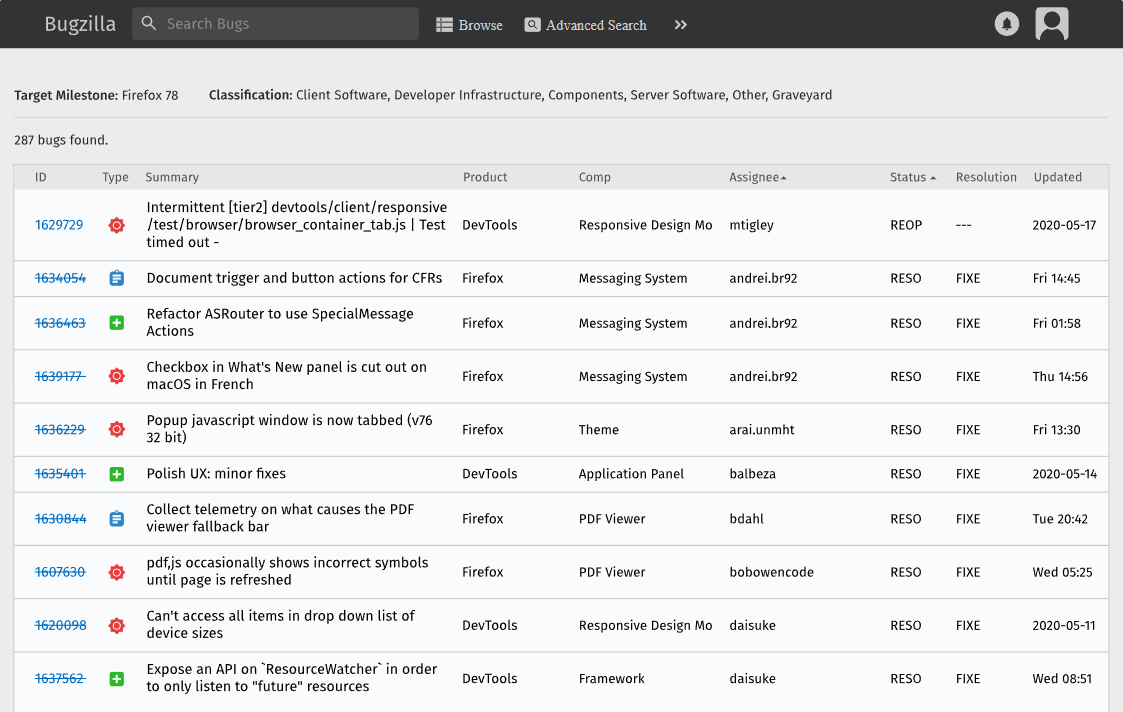


Рисунок 1.2.1 – Внешний вид блока веб-приложения «Bugzilla»

Функционал приложения включает типичный для систем такого рода набор инструментов, необходимых для контроля над ошибками в проектах:

* создание новых дефектов: пользователи могут создавать новые записи о дефектах, указывая приоритет, важность, компонент, версию продукта;
* назначение ответственных: каждому из дефектов может быть присвоен ответственный за его исправление
* отслеживание статуса: каждому из дефектов может содержать информацию об исправлениях и их состоянии;
* создание отчетов: программа может предоставлять отчеты и статистику по дефектам для анализа и принятия управленческих решений;
* поиск и фильтрация: предоставляет возможности поиска и фильтрации дефектов для быстрого доступа к нужной информации;

Однако следует отметить, что установка Bugzilla может оказаться достаточно трудоемкой из-за зависимости от модулей Perl. Также взаимодействие с ПП может быть затруднено из-за сложности администрирования или непонятно интуитивно интерфейса — наиболее четко это прослеживается при пользовании приложением для ОС компьютера (см. рисунки 1.2.1 – 1.2.2) .

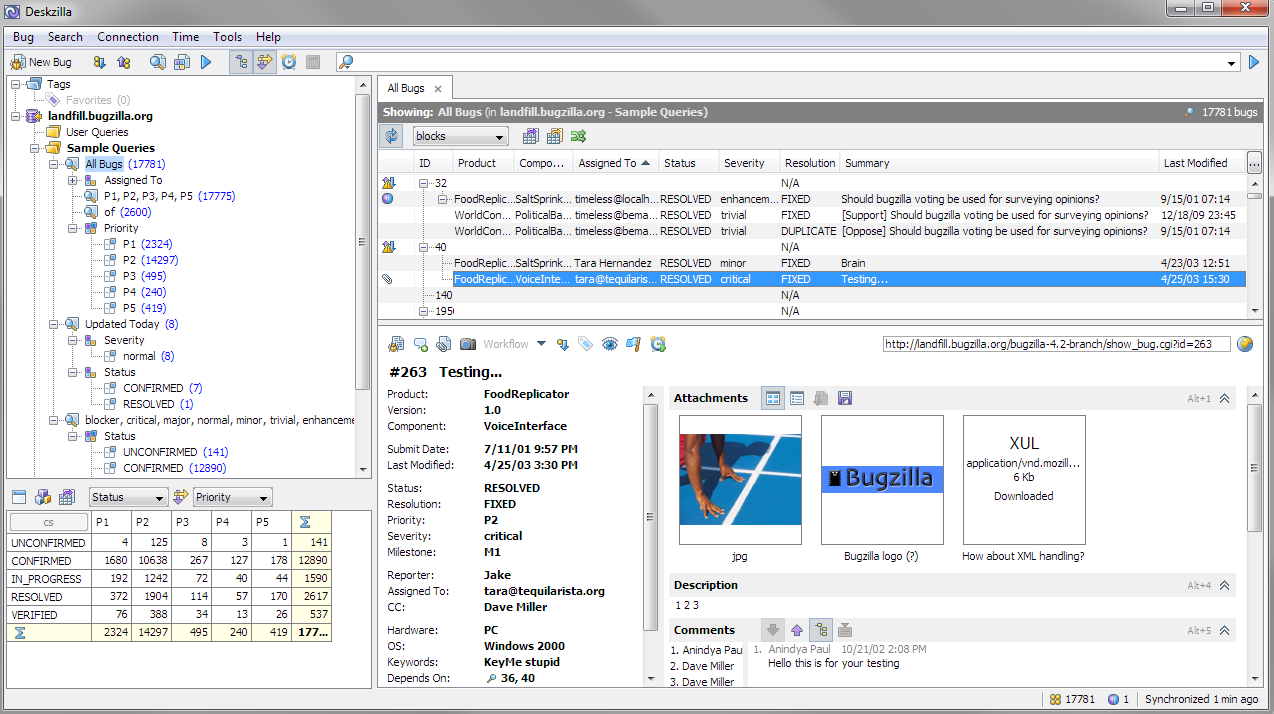


Рисунок 1.2.2 – Внешний вид блока программы «Bugzilla»

Для работы Bugzilla необходимы Apache, Perl и БД MySQL, что требует отдельной настройки и установки и усложняет процесс внедрения самой системы.

2. JIRA – система управления проектами и отслеживания задач, разработанная компанией Atlassian [6]. В настоящее время система отслеживания дефектов JIRA является одной из самых новых и популярных среди представленных. Она обладает наиболее широкой, можно сказать избыточной, функциональностью среди систем отслеживания. В целом JIRA повторяет архитектуру Bugzilla и обладает схожим функционалом. Далее опишем процесс отслеживание дефектов в данной системе:

Создаваясь, отчет о дефекте отправляется ответственному за реализацию данного функционала, создается так называемая «петля разработчик – тестировщик», в которой происходит цикличная переадресация сообщений. Любой из субъектов может перенаправить сообщение по собственному усмотрению.

Каждой существующей проблеме можно поставить приоритет важности, адресовать на себя, добавить комментарий. Это может быть как общий комментарий, так и комментарий направленный одному человеку, это используется, когда ведущий разработчик переадресует сообщение своему коллеге, указывая какую-то техническую подробность, которая нужна только ему. Сообщению можно установить статус «в начале работы», и соответственно указать, когда работы над ним закончены. Система поддерживает возможность создания персонифицированных сообщений. Вся информация может быть предоставлена пользователям в виде временных отрезков, нанесенных на общую временную шкалу (см. рисунок 1.2.3).

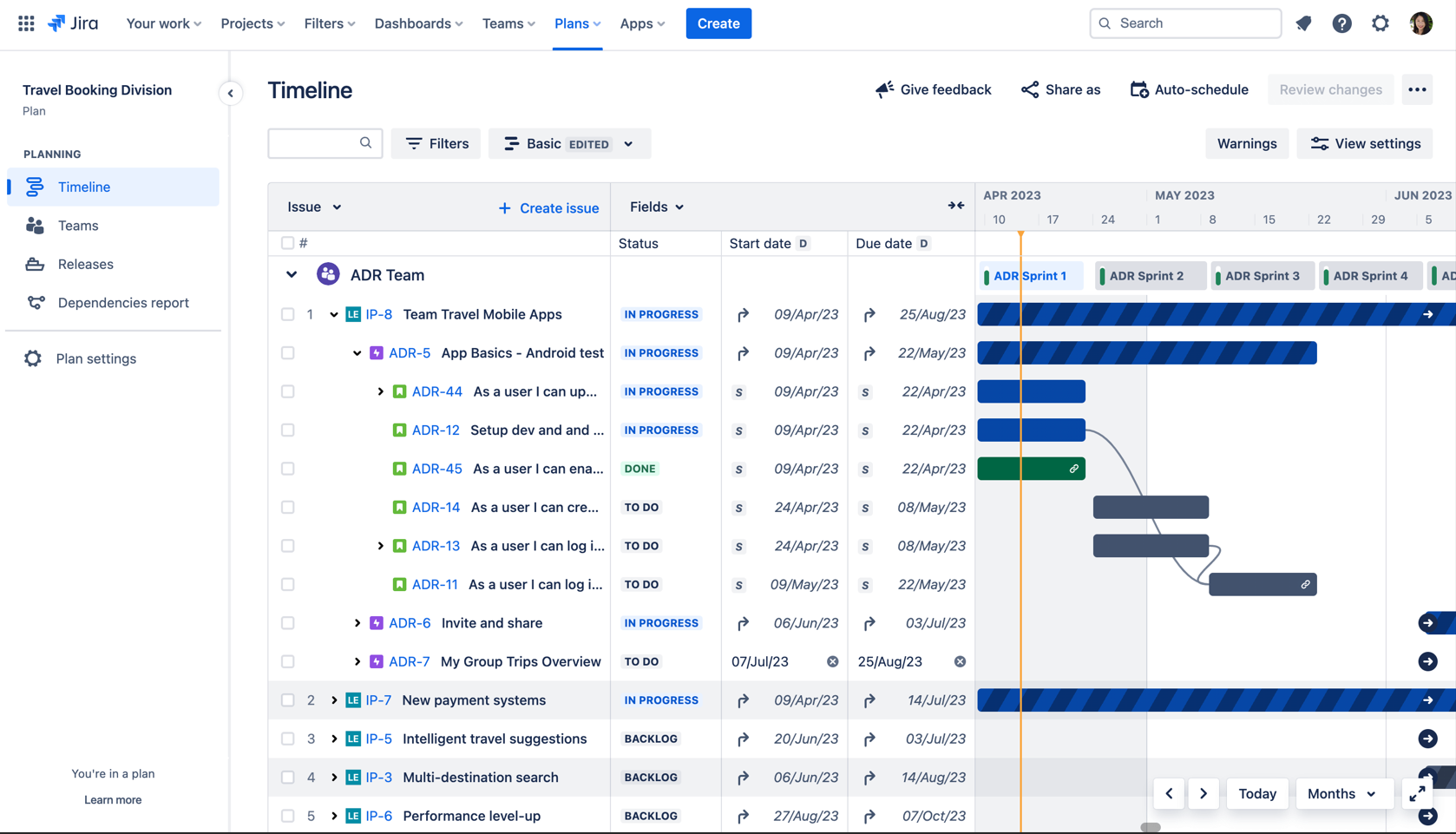


Рисунок 1.2.3 – Внешний вид блока системы «JIRA»

Система JIRA имеет избыток функциональных возможностей, что может быть проблемой для руководителей проектов без достаточного опыта. Вследствие этого сотрудники вынуждены затрачивать значительное количество времени на освоение функционала системы и его разбор.

Система обладает существенным недостатком — она платная. Стоимость установки JIRA на один сервер начинается от 4000 белорусских рублей. Данное обеспечение подходит для крупных компаний с большим штатом тестировщиков и крупным денежным оборотом.

3. Trac – это открытое ПО, являющееся одновременно инструментом для управления проектом и системой отслеживания ошибок. Разработка проекта Trac ведется компанией Edgewall Software и распространяется по лицензии Modified BSD [7], что предоставляет возможность для написания собственных модулей (далее – плагинов) и внедрения их в Trac. Интерфейс Trac представляет собой набор полей и текстовых блоков (см. рисунок 1.2.4), хотя по мнению сообщества он несколько отстает от современных эргономических требований, имеется возможность модернизировать его плагинами из общего доступа или написанными самостоятельно. Система использует в работе SVN репозиторий, так что использовать его имеет смысл только вместе с SVN.

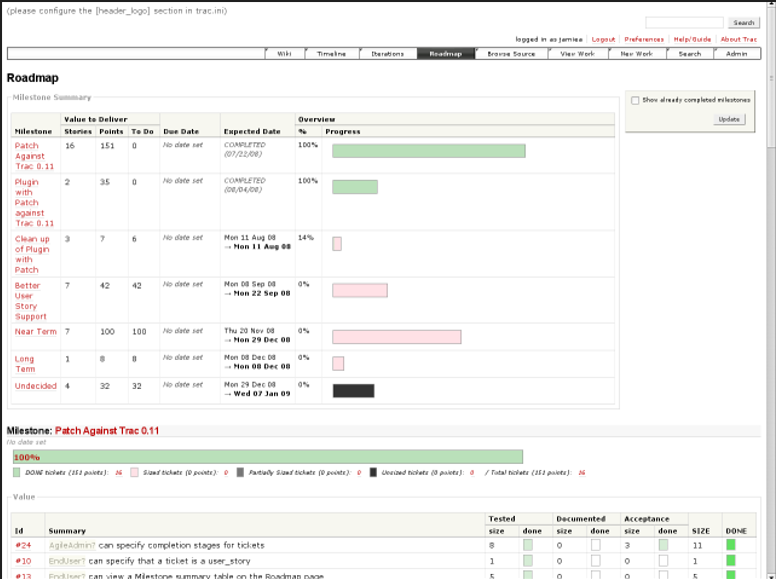


Рисунок 1.2.4 – Внешний вид блока системы «Trac»

Система имеет возможность разделение проекта на этапы контроля и выполнения, все изменения по проекту заносятся на временную шкалу. Trac был написан на Python и является кроссплатформенной системой. Эта система подходит для разработчиков, желающих внедрить комплексное решение для управления проектами и отслеживания ошибок, а так же для тех кто хочет модифицировать систему самостоятельно, дописав к нему дополнительный функционал.

4. The Bug Genie – кроссплатформенная система, написана на PHP. Проект достаточно успешно развивался, последняя версия вышла в марте 2008. The Bug Genie предоставляет базовый набор инструментов для регистрации ошибок, управления приоритетами и формирования задач для разработчиков. Система также предоставляет возможность оповещать всех заинтересованных разработчиков о появлении ошибки.

Система отслеживает ошибки в зависимости от версии и конфигурации ПО. Все ошибки сохраняются в единую БД, представляющую собой базу знаний об ошибках в проекте. Далее по этой БД формируется отчетность (см. рисунок 1.2.5). Однако данное программное решение не позволяет успешно интегрировать себя в окружение, не предусмотренное программистом.

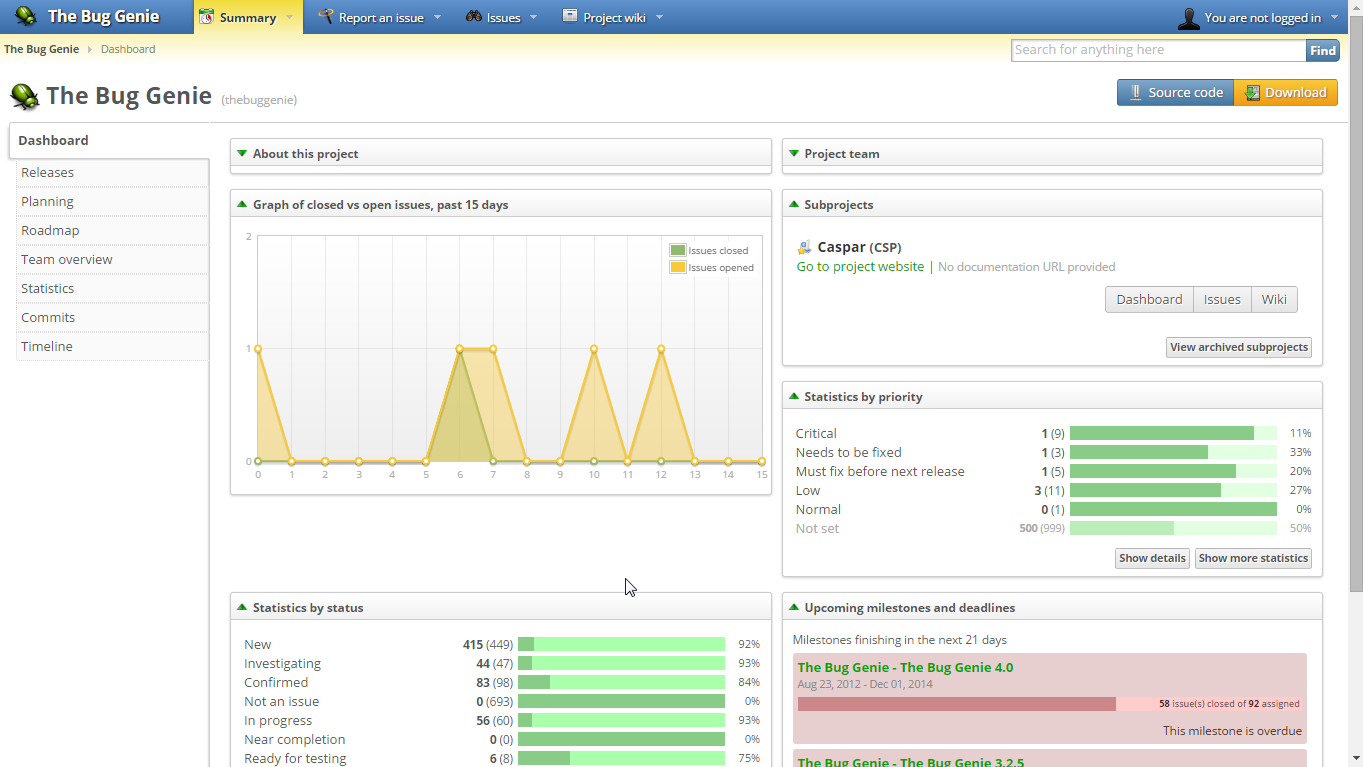
****

Рисунок 1.2.5 – Внешний вид сводки по проекту в «The Bug Genie»

Интеграция с другими инструментами серы разработки является ключевым аспектом при выборе системы отслеживания дефектов. В данном контексте Bugzilla и Trac предлагают относительно ограниченные возможности интеграции по сравнению с JIRA. JIRA, в свою очередь, предлагает широкие возможности интеграции с различными средствами разработки, такими как системы контроля версий (например, Git и менее популярной SVN), инструменты непрерывной интеграции и другими.

Следующим важным критерием является тип лицензии. Bugzilla и Trac являются свободным программным обеспечением, в то время как JIRA является платным. Это может оказать влияние на решение о выборе системы в зависимости от финансовых возможностей компании. Также следует учитывать язык программирования и зависимости каждой системы. Например, Bugzilla требует установки Perl, Apache и MySQL, что может потребовать дополнительных усилий при настройке и поддержке.

Современный эргономичный ГПИ снижает порог вхождения для пользователей и сильно влияет на принятие решения в процессе начального выбора ПП [8], большинство из аналогов СОДПП имеют некоторые затруднения в выдерживании конкуренции при рассмотрении данного аспекта, на что указывает повышенный спрос на платные продукты и доминирующая позиция проекта JIRA.

Интерфейсы систем схожих с разрабатываемой несколько перегружены вследствие специфики предметной области, но уровень перегруженности существенно разнится в зависимости от даты выпуска программы. С течением времени эргономические требования и общие тенденции в дизайне сильно поменялись. Так же от даты выпуска зависит и формат приложения — большинство современного ПО для отслеживания дефектов программных продуктов являются веб-приложениями, или веб-приложениями, которые были интегрированы в ОС компьютера. Такие приложения способны функционировать только при условии стабильного подключения к интернету, так как БД находятся на удаленных серверах.

Освояемость программного продукта тесно связана с его дизайном. Дизайн ПП оказывает прямое влияние на процесс освоения, определяя уровень доступности функционала, ясность ГПИ и интуитивную понятность работы программы. Хорошо спроектированные программные продукты демонстрируют более высокий уровень освояемости, что, в свою очередь, способствует повышению их эффективности использования и удовлетворенности пользователей [9]. Эта характеристика играет ключевую роль в успешной адаптации пользователей к новым программным средствам и определяет скорость достижения оптимальной производительности при их использовании.

Основываясь на приведенных данных, составим таблицу 1.2.1 — таблицу сравнения аналогов СОДПП.

Таблица 1.2.1 – Сравнение аналогов СОДПП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Bugzilla | JIRA | Trac | The Bug Genie |
| Тип лицензии | Свободное ПО | Платное | Открытое ПО | Открытое ПО |
| Язык программирования | Perl | Java | Python | PHP |
| Зависимости | Perl, Apache, MySQL | Java | SVN | PHP |
| Интерфейс | Отстающий | Современный | Отстающий | Устаревший |
| Освояемость | Средняя | Низкая | Низкая | Высокая |
| Сложность установки | Высокая | Средняя | Средняя | Средняя |
| Подходит для маленьких компаний | Нет | Нет | Да | Нет |
| Интеграция с другими инструментами | Возможна | Возможна | Возможна | Ограничена |

Таблица отражает основную динамику, существующую на современном рынке, и отображает основные его тенденции.

В данном подразделе были приведены лишь некоторые, самые широко известные, ПС по отслеживанию ошибок в программных продуктах. Например, не были затронуты некоторые приложения, которые используются в целях решения задач по отслеживанию дефектов, но не были разработаны для них. Модульные системы, встроенные в какое-либо приложение, целью которого не является отслеживания дефектов (например – GitHub Issues) так же не рассматривались, хотя и пользуются некоторым спросом.

Фактически для отслеживания дефектов может использовать любое существующее средство связи и инструмент, способный содержать в себе несколько вложенных списков — такой функционал может быть реализован даже непрограммно.

Проанализировать эффективность таких видов отслеживания средств невозможно ввиду их специфического назначения и особенностей функционирования. Как уже отмечалось, эти средства могут быть эффективными в определенных сценариях использования, однако их функциональность и возможности могут быть ограничены по сравнению с специализированными инструментами для отслеживания дефектов.

Кроме того, важно учитывать, что эффективность отслеживания дефектов в значительной степени зависит от организации процесса работы команды, внедрения подходящих методологий разработки и использования соответствующих инструментов управления проектом. Даже самая передовая система отслеживания дефектов может оказаться неэффективной, если ее не используют правильно или если в ней отсутствует поддержка и вовлеченность команды.

Таким образом, при выборе средства для отслеживания дефектов необходимо учитывать не только его функциональные возможности, но и контекст использования, особенности проекта и предпочтения команды разработчиков.

****1.3 Выводы и постановка задач на дипломное проектирование****

Рассмотрев существующие аналоги СОДПП и предметную область системы можно обозначить общие пожелания и запросы пользователей к системе, сделать выводы о назначении системы, сформировать ее задачи и функционал.

В частности, из анализа предметной области можно сделать вывод, что СОДПП включает в себя средства налаживания межличностных коммуникаций между тремя основными типами пользователей, разделенных по критерию принадлежности к определенной должности: тестировщик, программист и менеджер проекта. Однако программный продукт в основном используется лишь одним из представленных специалистов — тестировщиком, вследствие чего и будем рассматривать его как рядового пользователя продукта. Существует необходимость выделения другого типа пользователя, способного отслеживать общий прогресс команды разработчиков в реальном времени и вносить правки во все учитываемые ПП — обычно данная возможность необходима менеджеру проекта, но в зависимости от внутреннего распорядка компании может затрагивать и других специалистов. Будем называть такого пользователя «привилегированным», вне зависимости от занимаемой им должности и степени важности для него этой возможности. Итого получаем две роли пользователей в системе:

а) *Рядовой пользователь*: сотрудник отдела тестирования ПО или разработчик, относящийся к реализации части дефектного функционала ПО.

б) *Привилегированный пользователь*: менеджер проекта или человек любой другой должности, ответственный за отчетность о проекте.

Хоть данные роли и имеют очень схожие полномочия в СОДПП, но привилегированный пользователь отличается возможностью наблюдать за деятельностью всех работников, ведущих учет в системе. Разделение на пользователей не является обязательным на этапе предоставления базового программного решения, а в основном необходимо для демонстрации возможности дальнейшего разделения функционала и примера внедрения такового. Базовое решение обычно ограничивается минимальным необходимым функционалом. Выделим его в рамках разрабатываемой системы и коротко опишем каждую из функций системы:

1. Регистрация аккаунта — выполнение процедуры, при которой пользователь заполняет электронную форму, содержащую поля "Имя пользователя" и "Пароль";
2. Авторизация — идентификация и аутентификация существующего пользователя;
3. Добавление ПП — добавление ПП в личный список для дальнейшего учета. Присваивание ПП названия и существующих дефектов данного продукта;
4. Архивирование ПП — сокрытие ПП из общего списка отслеживаемых продуктов и помещение их в отдельный список с разрешения привилегированного пользователя.
5. Редактирование ПП — редактирование названия ПП, текстового описания, список работников, ответственных за исправление дефекта;
6. Добавление дефектов — установка названия дефекта, его важности, приоритета, результата, ожидаемого результата, описания найденного дефекта, комментария составителя и статуса;
7. Изменения состояния существующего дефекта — пометка дефекта как исправленного или изменения данных о нем;

В ходе разработки системы необходимо предусмотреть возможность легкой и не слишком трудоемкой подстройки под нужды потребителя. То есть необходимо разработать модульное решение, демонстрирующее возможности легкой модификации и предоставить интерфейсы, способные согласовать систему не только с программным окружением, но и с потенциальными функциональными модулями, призванными эту систему расширить.

Следует отметить, что у пользователей СОДПП могут быть несколько завышенные требования к системе, ввиду своей сильной включенности в сферу программного обеспечения и углубленного понимания общих практик (стандартов) в индустрии его разработки.

1. ЭРГОНОМИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ
   1. Разработка алгоритмов работы пользователя

Для проведения эргономической разработки системы необходимо составить спецификацию функций системы и произвести разбор их содержимого в рамках данного дипломного проекта.

Минимальный необходимый функционал базового решения не является исчерпывающим в контексте современных требований к функциональному наполнению системы. Вследствие этого, дополнительно расширим функциональность, раскрыв и разбив на части общие функции, которые были выделены ранее.

Напомним, что полномочия пользователей различается только в объеме доступной к обработке информации, что не затрагивает функциональность в целом. То есть, состав функций для обоих пользователей является одинаковым, хотя в одной и той же функции отображается разное количество отслеживаемых проектов и их дефектов.

На основе перечная функций в разделе 1.3 пояснительной записки, выполним анализ содержания основных функций СОДПП.

1. Регистрация пользователей предполагает выполнение процедуры, при которой пользователь заполняет электронную форму, содержащую поля «Login» и «Password»;
2. Идентификация пользователей реализуется путем поиска в базе данных сервиса информации, соответствующей введенному пользователем имени;
3. Аутентификация пользователей осуществляется сравнением значения, полученного в результате считывания введенного пароля, или его зашифрованным значением, хранящимся в системе и связанным с введенным именем пользователя;
4. Авторизация пользователей выполняется путем определения их роли на основе данных, хранящихся в системе ПП;
5. Добавления новых программных продуктов осуществляется путем нажатия пользователем предназначенной для этого кнопки, ввода названия ПП, его описания, и пользователей, способных взаимодействовать с данным продуктом;
6. Добавление новых дефектов производится путем нажатия пользователем кнопки добавления в окне (на странице) списка дефектов существующего проекта, и дальнейшего ввода данных о нем: краткого описания в соответствующее поле, ожидаемого результата, описания, выбора приоритета и важности дефекта, в появляющемся окне;
7. Просмотр списка отслеживаемых дефектов производится путем пролистывания страницы списка, после нажатия на название проекта или поиска по таблице, содержащей данные о состоянии дефектов и проектов, на главном экране приложения;
8. Просмотр информации об отслеживаемом дефекте: истории прошлых действий, относящихся к какому либо, из внесенных в систему дефектов (далее – логов), текущего статуса, описания, важности и приоритета, осуществляется путем ознакомления с данными, выводимыми на экран в блоке с карточками дефектов в списке дефектов проекта;
9. Установка приоритетов отслеживаемых дефектов происходит путем выбора из нескольких предложенных вариантов при создании или изменении дефекта;
10. Установка важности отслеживаемых дефектов совершается выбором одного из предложенных вариантов в выпадающем списке при создании или изменении дефекта;
11. Изменение общей информации о дефекте или проекте осуществляется путем нажатия кнопки «Edit» в верхней части блока необходимого отслеживаемого объекта;
12. Архивирование дефектов или проектов происходит путем нажатия на кнопку «Archive» возле названия соответствующего объекта;
13. Установка текущего статуса дефекта происходит путем нажатия пользователем системы кнопки с надписью, советующей названию одного из этапов исправление дефектов ПО напротив названия этого дефекта;
14. Просмотр заархивированных проектов осуществляется путем выбора пользователем одного из возможных положений переключающего элемента («Active» и «Archived») в окне с таблицей данных о проектах;
15. Просмотр списка заархивированных дефектов предполагает раскрытие пользователем выпадающего списка внизу страницы существующего в системе проекта;
16. Поиск по текстовым полям дефектов подразумевает ввод ключевых слов поиска на странице таблицы дефектов и нажатия клавиши «Enter»;
17. Фильтрация дефектов по важности и приоритету происходит путем выбора необходимых значений этих характеристик объекта в верхней части страницы таблицы дефектов и нажатия кнопки «Apply Filters»;
18. Просмотр справочной информации о системе осуществляется путем нажатия пользователем на надпись «About» в верхней части интерфейса программы;
19. Выход из текущей сессии пользователя производится по нажатию пиктограммы выхода в правой верхней части любой из страниц приложения;
20. Полное закрытие программы и автоматическое закрытие сессии производится путем закрытия окна программы.

Принимая во внимание содержание основных функций и специфику СОДПП, проведем анализ функций системы и распределим их между человеком и компьютерными средствами, то есть распределим их между исполнителями. Результаты предоставим в виде таблицы 2.1.1

Таблица 2.1.1 – Распределение функций между исполнителями

|  |  |  |
| --- | --- | --- |
| Название функции | Действия человека в системе | Операции компьютера в системе |
| 1 | 2 | 3 |
| 1. Регистрация пользователя в системе | Вводит логин и пароль в соответствующие поля | Произведение записи введённых данных в систему |
| 2. Идентификация пользователя | Вводит логин | Сверка введенного и сохраненного значений |
| 3. Аутентификация пользователя | Вводит пароль | Сверка введенного и сохраненного значений |
| 4. Авторизация пользователя | — | Допуск пользователя к системе или выдача сообщений об ошибке |
| 5. Добавления новых программных продуктов в систему | Вводит необходимые данные | Обработка и запись введённых данных о ПП в файлы системы |

Продолжение таблицы 2.1.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 6. Добавление новых дефектов ПП в систему | Щелчок\* по кнопке добавления на странице списка дефектов существующего проекта, и вводит данные о нем | Создание сущности дефекта в файлах системы, согласование ее с существующим проектом и создание связи между ними. Выведение сообщения об успешном добавлении дефекта в интерфейс системы |
| 7. Просмотр списка отслеживаемых дефектов | Просматривает выведенные данные | Выведение информацию о дефектах в виде столбца карточек (далее – списка дефектов) |
| 8. Просмотр информации об отслеживаемом дефекте | Просматривает выведенные данные | Предоставление доступа к данным о дефекте, включая историю действий, текущий статус, описание, важность и приоритет. Отображает данных на экране ПК, обновление при изменении данных |
| 9. Установка приоритетов отслеживаемых дефектов | Выбирает опцию из выпадающего списка | Отображение перечня уровней приоритетности решения дефекта и сохранение введенного значение в системе |
| 10. Установка важности отслеживаемых дефектов | Выбирает опцию из выпадающего списка | Предоставление вариантов уровней важности дефекта и сохранение введенного значение в системе |

Продолжение таблицы 2.1.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 11. Изменение общей информации о внесенном дефекте или проекте (далее – объекте системы) | Щелчок по кнопке «Edit» в верхней части блока отслеживаемого объекта системы | Запись обновленных данных в файловую систему. Выведение сообщение об успешном изменении данных объекта |
| 12. Архивирование объектов системы | Щелчок по кнопке «Archive» напротив названия объекта | Удаление объекта из списка активных и добавление в список архивированных в файловой системе. Удаление объект из текущего окна ГПИ |
| 13. Установка текущего статуса дефекта | Щелчок по кнопке изменения статуса дефекта | Обновление информации в файловой системе и изменение текущего окна ГПИ в соответствии с ними |
| 14. Просмотр заархивированных проектов | Щелчок на переключающий элемент ГПИ в головке таблицы проектов | Фильтрация типа отображаемых проектов на стороне пользовательского клиента и обновление таблицы проектов в соответствии с ней |
| 15. Просмотр списка заархивированных дефектов | Щелчок по пиктограмме стрелки возле раскрывающегося списка | Отображение информации о заархивированных дефектах. Выведение в интерфейс заранее загруженной системой информации |

Продолжение таблицы 2.1.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 16. Поиск по текстовым полям дефектов | Ввод текстовой информации в форму поиска и нажатие клавиши «Enter» или нажатие соответствующей кнопки ГПИ | То же |
| 17. Фильтрация дефектов по важности и приоритету | Выбор одной из опций выпадающего списка и нажатие по ней. Нажатие кнопки «Apply Filters» | » |
| 18. Просмотр справочной информации о системе | Щелчок по надписи «About» в верхней части интерфейса программы | » |
| 19. Выход из текущей сессии пользователя | Щелчок по пиктограммы выхода в правой верхней части любой из страниц приложения | Инициирование процесса завершения сеанса пользователя после подтверждения выхода. Предоставление информации о последствиях выхода из сессии |
| 20. Полное закрытие программы и автоматическое закрытие сессии | Щелчок по кнопке «X» в правом верхнем углу окна или другое действие, предусмотренное ОС | Закрытие текущей сессии пользователя. Завершение всех активных процессов и очистку временных данных |
| \*В данном контекста слово «щелчок» используется как обозначающие нажатие ЛКМ после наведения курсора мыши на элемент ГПИ, отображаемого на экране ПК пользователя. | | |

Распределение функций СОДПП сделано на основе принципов преимущественных возможностей и принципа взаимного дополнения СЧКС, ввиду своих архитектурных особенностей. В результате проведенного анализа можно сделать следующий вывод: ввиду идентичности функционала пользователей проектируемая система будет состоять из одной подсистемы «пользователь — ПК — среда», хоть и формально типов пользователей в системе два.

В результате проведенного анализа и распределения выполнения действий функций между человеком-оператором и ПК в проектируемой СЧКС можно сделать следующие выводы [10]:

– функции распределены между человеком и компьютером по функциональным возможностям и степени эффективности выполнения действий;

– для проектирования системы достаточно одного типа пользователя;

– выводы о структуре системы.

Разработаем алгоритм работы пользователя с учетом всей перечисленной функциональности. Типичный алгоритм работы пользователя в системе включает в себя большинство из функций выделенных ранее. Представим алгоритм в виде таблице 2.1.2.

Таблица 2.1.2 – Алгоритм работы пользователя

|  |  |  |
| --- | --- | --- |
| Содержание операции | Обращение к органам управления | Обращение к отображению ГПИ |
| 1 | 2 | 3 |
| Переход на окно входа в приложение | Запуск приложения путем щелчка на файл запуска программы | Отображение начального окна приложения |
| Идентификация и аутентификация пользователя в системе | Заполнение текстовых полей формы путем взаимодействия со средствами текстового ввода ПК (далее – клавиатурой) | Вывод на экран формы входа пользователя. Выведение уведомления об ошибке при некорректно введенных данных |

Продолжение таблицы 2.1.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Добавления нового программного продукта в систему | Щелчок ЛКМ по кнопке добавления ПП на экране. Взаимодействие пользователя с клавиатурой | Отображение полей для ввода текстовых данных |
| Подтверждение добавления ПП в систему | Щелчок ЛКМ по кнопке добавления подтверждения ввода на экране | Открытие страницы пустого списка дефектов нового ПП |
| Инициация добавления новых дефектов ПП в систему | Щелчок ЛКМ по кнопке добавления дефекта на экране | Отображение пустой формы добавление дефекта в ГПИ системы |
| Заполнение текстовой информации о вносимом в систему дефекте | Взаимодействие пользователя с клавиатурой | Динамичное отображение вводимых в тестовые поля изменяющихся данных |
| Выбор приоритетности нового дефекта | Щелчок ЛКМ по кнопке одной из опций перечня приоритетов на экране | Отображение и динамическое изменение вида опций перечня |
| Выбор важности  нового дефекта | Щелчок ЛКМ по кнопке одной из опций перечня важности на экране | То же |
| Подтверждение добавления дефекта в список дефектов ПП | Щелчок ЛКМ по кнопке подтверждения добавления дефекта на экране | Сокрытие формы добавления дефекта. Отображение списка дефектов ПП и содержащихся в нем дефектов, а так же информации о них |

Продолжение таблицы 2.1.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Изменение статуса активного дефекта | Щелчок ЛКМ по кнопке на экране с названием статуса, следующего за текущим в цикле исправления дефектов (разработки) | Изменение индикатора текущего статуса дефекта. Изменение текста кнопки изменения статуса |
| Архивирование дефекта | Щелчок ЛКМ по кнопке архивирования дефекта на экране | Сокрытие объекта из списка активных |
| Архивирование ПП | Щелчок ЛКМ по кнопке архивирования проекта на экране | Сокрытие страницы просмотра ПП. Отображение таблицы заархивированных ПП |
| Просмотр списка активных ПП | Щелчок ЛКМ по переключающему элементу ГПИ | Сокрытие таблицы заархивированных ПП. Отображение таблицы активных ПП |
| Просмотр отслеживаемых дефектов | Наведение курсора на название существующего ПП и нажатие ЛКМ | Вывод в ГПИ списка отслеживаемых дефектов данного ПП |
| Поиск по текстовым полям дефектов | Нажатие на поле ввода ГПИ и ввод искомого текста путем взаимодействия с клавиатурой ПК | Изменение состава списка отслеживаемых дефектов |
| Просмотр информации об отслеживаемом дефекте | Наведение курсора на название требуемого дефекта и нажатие ЛКМ | Вывод данных о выбранном пользователем дефекте в ГПИ |
| Изменение информации об отслеживаемом дефекте | Нажатие ЛКМ по кнопке изменения дефекта пользователем. Взаимодействие с клавиатурой | Динамичное отображение вводимых в тестовые поля изменяющихся данных |

Продолжение таблицы 2.1.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Подтверждения внесенных в дефект изменений | Нажатие ЛКМ по кнопке подтверждения внесения изменений на экране | Сокрытие страницы изменения дефекта. Отображение таблицы всех активных дефектов в системе |
| Инициация изменения информации о ПП в системе | Нажатие ЛКМ по пиктограмме стрелки. Нажатие ЛКМ по опции изменения информации о ПП в системе. | Отображение формы изменения ПП в ГПИ системы |
| Изменения текстовой информации о ПП в системе | Взаимодействие пользователя с клавиатурой | Динамичное отображение вводимых в тестовые поля изменяющихся данных |
| Сохранение внесенной измененной информации о ПП | Нажатие ЛКМ по кнопке сохранения внесенных изменений на экране | Сокрытие формы изменения информации о ПП. Отображение таблицы со всеми активными проектами в системе |
| Полное закрытие программы и закрытие сессии | Нажатие ЛКМ после наведения на кнопку с символом «Х» в правом верхнем углу экрана окна приложения | Скрывает объекты, относящиеся к системе из текущего вида интерфейса ОС |

Для формирования четкого представления об алгоритме работы пользователя и более краткого его описания, представим его полный, более общий, вариант графически (в виде блок-схемы) на рисунке 2.1.1 и отобразим на нем все возможные из функций системы.

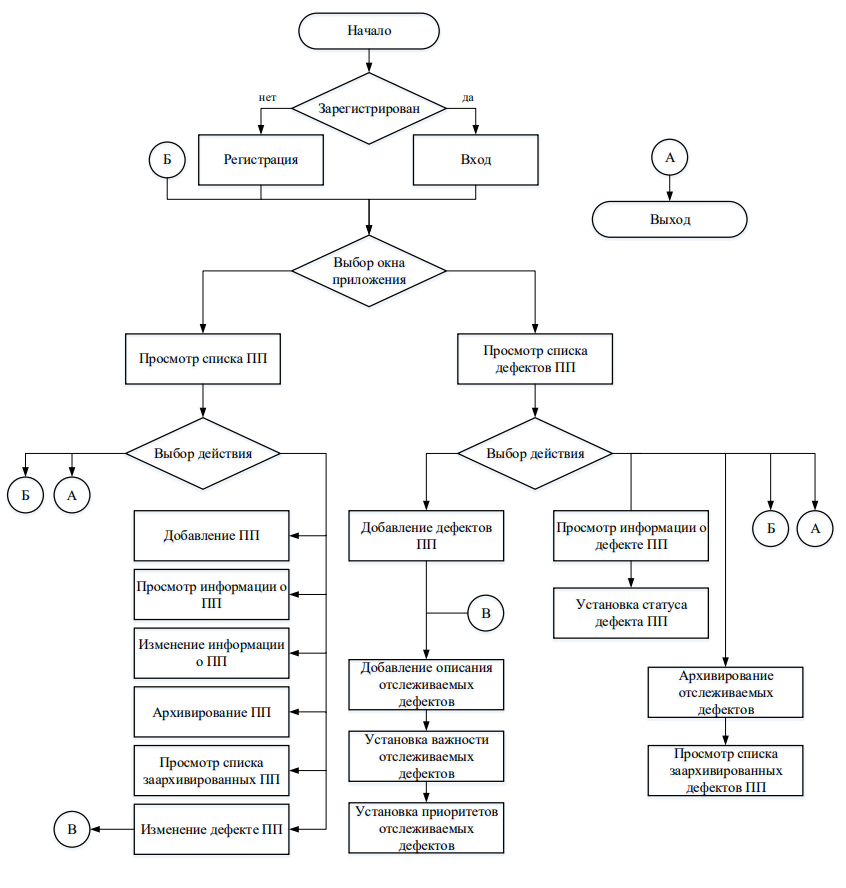
****

Рисунок 2.1.1 – Алгоритм работы пользователя в виде блок-схемы

2.2 Разработка эргономических требований и сценария информационного взаимодействия

Эргономические требования к СЧКС – это требования к системе в целом, ее отдельным подсистемам, оборудованию, рабочей среде, а также человеку-пользователю компьютера, определяемые свойствами человека и устанавливаемые для обеспечения его эффективной и безопасной деятельности.

Следует заметить, что формулирование эргономических требований необходимо для того, чтобы обеспечить высокий уровень эргономичности системы [10].

Поскольку перечень ЭТ может быть достаточно обширным, его разделяют на 6 групп в соответствии с принятой в инженерной психологии и эргономике классификацией. При этом выделяют:

1. Антропометрические требования определяются анатомическими, морфологическими и биомеханическими свойствами человека;
2. Физиологические требования учитывают энергетические и скоростные возможности человека;
3. Психофизиологические требования обусловлены возможностями и особенностями сенсорных систем человека;
4. Психологические требования определяют соответствие СЧКС и ее элементов психологическим особенностям человека;
5. Гигиенические требования определяют безопасные условия работы человека;
6. Социально-психологические требования определяют соответствие конструкции машины и организации рабочих мест характеру и степени группового взаимодействия.

Каждое из требований входит в состав отдельного свойства системы. Перечислим свойства, относящиеся к разрабатываемой системе, и дадим их краткую характеристику:

– управляемость: оценивает эффективность выполнения функций системы в условиях, удобных для человека, как для оператора системы;

– освояемость: оценивает время, необходимое для освоения возможностей самой системы.

Очевидно, что в рамках данного проекта, как и случае разработки любого другого ПП, необходимо составить и придерживаться списка эргономических требований, влияющих на работу пользователя системе. Требования, разработанные для СОДПП, предоставлены в таблице 2.2.1

Таблица 2.2.1 – Общие эргономические требования проектируемой системы

|  |  |  |
| --- | --- | --- |
| Эргономические показатели | | Эргономические свойства |
| Единичные | Групповые |  |
| 1 | 2 | 3 |
| - соответствие размеров знаков на экране дисплея оперативному порогу зрения человека [10];  - соответствие контраста знаков и фона оптимальным условиям восприятия [11];  - соответствие вида контраста знаков и фона уровню освещенности экрана;  - яркость и размер элементов управления оптимального считывания;  - отсутствие требований, связанных с управляемостью системы, несоответствующих возможностям органов чувств человека;  - обозначение недоступных или неактивных элементов управление ГПИ хорошо различимым блеклым цветом;  - использование пролистываемых и  раскрывающихся списков в целях оптимизации экранного пространства; | Психофизиологический | Управляемость |

|  |  |  |
| --- | --- | --- |
| Продолжение таблицы 2.2.1 | | |
| 1 | 2 | 3 | |
| - обеспечение заметных отличий между органами управления, отвечающими за разные функции  [10];  - один и тот же характер действий на протяжении всего периода работы в системе в схожих ситуациях;  - наличие указаний на проблемы, возникающие в процессе управления системы;  - соответствие цветов знаков стереотипам восприятия [10];  - отсутствие неоднозначного толкования требований, инструкций и команд;  - соответствие количества одновременно предъявляемых сигналов возможностям внимания человека;  - наличие индикации степени выполнения заданий (операций) [10];  - минимизация сложности и количества шагов для выполнения основных операций управления; | Психологический | Управляемость | |

|  |  |  |
| --- | --- | --- |
| Продолжение таблицы 2.2.1 | | |
| - отсутствие требований, связанных с управляемостью системы, несоответствующих скоростным возможностям человека; | Физиологический | Управляемость |
| - соответствие шрифта стандартам, представленным в ГОСТ ЕН 894-2-2002 [12] | Психофизиологический | Освояемость |
| - соответствие сложности инструкций, предоставляемых обратной связью, времени, отводимому на их усвоение;  - исключение открытия ненужных окон [11];  - предоставление навигационных обзоров [11];  - все функциональные элементы интерфейса подписаны или имеют знаки, соответствующие их назначению;  - однотипное расположение схожих объектов на разных страницах [11]; | Психологический |
| Примечание: некоторые единичные показатели, приведенные в составе группового показателя, определяющего одно эргономическое свойство, оказывают влияние и на формирование других эргономических свойств. Но в данной таблице они не повторяются | | |

Стандарты устанавливают общие требования к разрабатываемому сервису, подчеркивая необходимость полного соответствия техническому заданию. Основной акцент делается на эргономике, включая яркость, контрастность и размеры знаков на экране для удобного восприятия информации. Важным также является организация пространства на экране, минимизация второстепенных объектов и оптимизированные параметры текста, которые распространяются не только на экран, но и на всю рабочую область.

Характеристики шрифта относятся к свойству «Освояемость», так как предполагается, что текст на органах управления воспринимается только в момент обучения коммуникации с системой, а в дальнейшем пользователь взаимодействует с ними по сформированному стереотипу.

Контраст должен соответствовать правилу: при наблюдениях предметов, угловые размеры которых не меньше 0,5°, минимальная величина контраста для человеческого глаза близка к 0,02. Размер определяется из расчета кратности числу два, минимальное возможное значение шрифта шесть пикселей.

Рекомендуемые высоты знаков обеспечиваются при угле наблюдения, равном 18'—22'; допустимые высоты знаков обеспечиваются при угле 15'—18'. Высота знака при угле менее 15' недопустима. Рекомендуемые размеры знаков могут быть рассчитаны приблизительно [12]:

– рекомендуемый диапазон ширины знака составляет от 60 до 80 % высоты знака;

– если поверхность индикатора закруглена, и угол зрения не является прямоугольным, то ширина знака должна составлять 80—100 % его высоты. Ширина знака менее 50% его высоты недопустима;

Анализ разрабатываемой системы и разработаны эргономические требования к СОДПП, в соответствии с действующими на момент написания стандартами [11][12].

В ход эргономического проектирования был создан один из вариантов ГПИ системы, удовлетворяющий составленным выше требованиям.

Рассмотрим сценарий информационного взаимодействия пользователя с системой, отразив характер и структуру содержания системы, а так же опишем логику самого взаимодействия.

Для входа в систему пользователь должен ввести логин и пароль, после чего нажать на кнопку «Sing In». В случае если пользователь ещё не зарегистрирован в системе, для пользования сервисом ему необходимо это сделать, нажав на кнопку «Sign Up».

****

Рисунок 2.2.1 – Блок программы «Вход в систему»

Для регистрации в СОДПП пользователь должен заполнить поля логина, пароля, соответствующего правилам, отображаемым в всплывающем элементе ГПИ, и поля подтверждения пароля, после чего повторно нажать на «Sign Up».

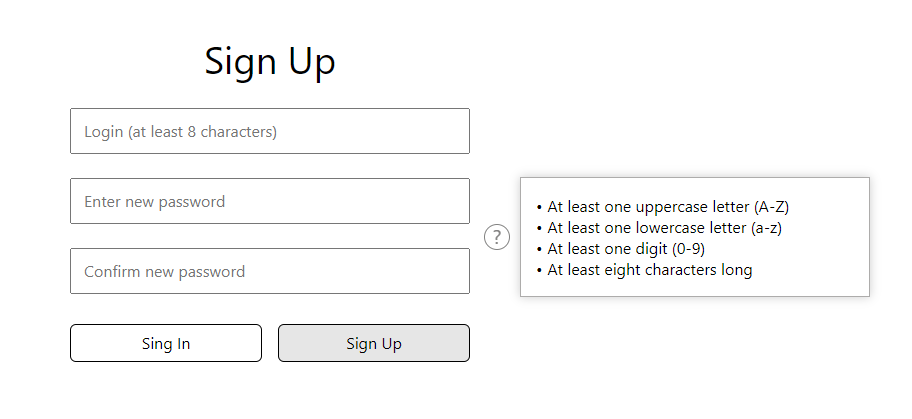


Рисунок 2.2.2 – Блок программы «Регистрация»

После успешного входа или регистрации пользователь попадает на главную страницу приложения, содержащую общую для всех проектов таблицу дефектов. На странице пользователь может произвести следующие действия, путями, описанными в подразделе 2.1:

1. Закрыть или свернуть приложение (данные действия возможны на любом этапе взаимодействия с системой);
2. Выйти из текущей сессии (возможно на любом этапе взаимодействия с системой);
3. Просмотреть список из 10 доступных пользователю дефектов, а так же перейти на следующую или предыдущую страницу просмотра таблицы, если это возможно;
4. Просмотреть краткую информацию о каждом из отображенных в ГПИ дефектов, а именно: статус, название проекта, приоритет дефекта, важность дефекта, список пользователей имеющих доступ к внесению изменений в данный дефект ПП;
5. Отфильтровать список отображаемых дефектов ПП, по какой либо из общих характеристик дефектов;
6. Произвести поиск по полному списку дефектов;
7. Выбрать одно из возможных действий в выпадающем меню «Actions», после нажатия на пиктограмму стрелки возле надписи:
   1. Изменить информацию о дефекте;
   2. Просмотреть полную информацию о дефекте;
   3. Просмотреть полную информацию о проекте (ПП) дефекта;
   4. Скрыть дефект из списка активных и добавить его в архив;
8. Перейти на страницу с кратким описанием системы;
9. Перейти на страницу с таблицей внесенных в систему ПП.

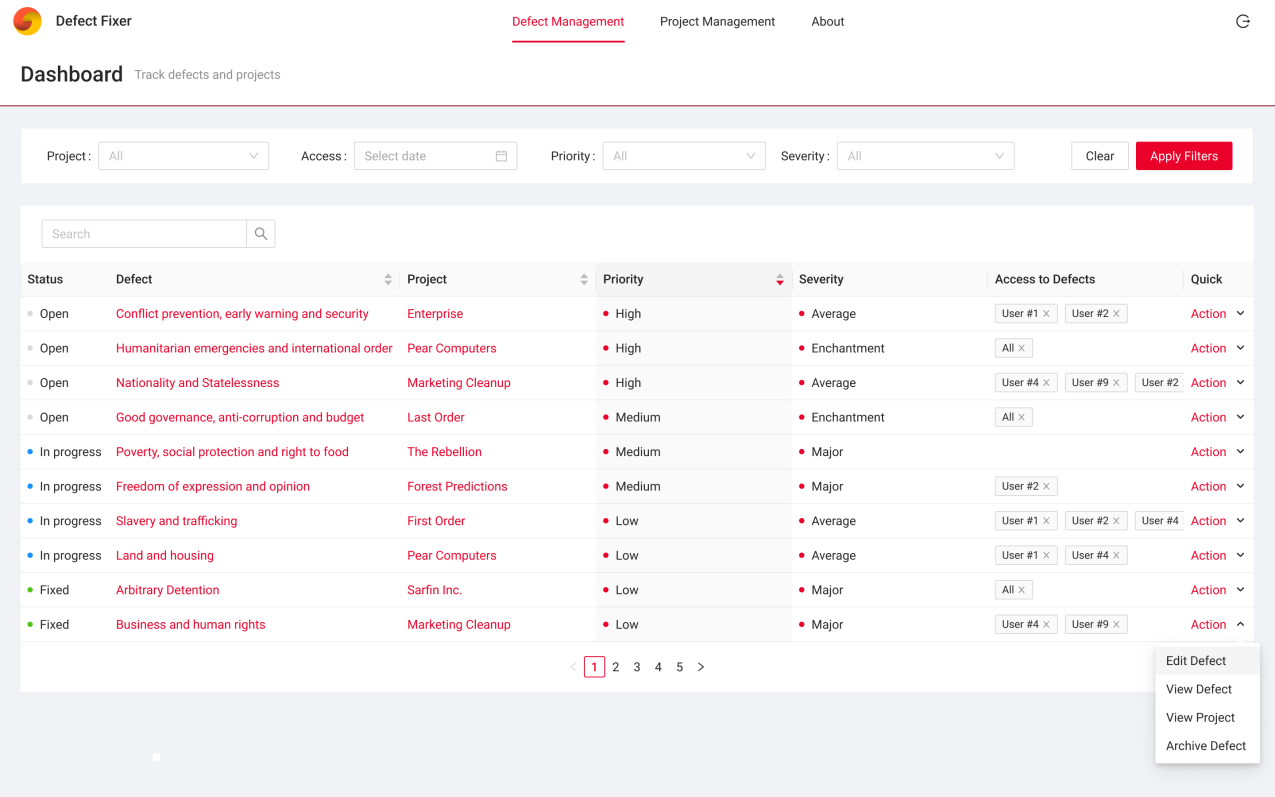


Рисунок 2.2.3 – Главная страница приложения

Нажав на название любого из дефектов, или выбрав опцию просмотра в выпадающем меню «Action», пользователь попадает на страницу просмотра списка дефектов проекта, с закрепленным в его верхней части выбранным ранее дефектом, где может:

1. Быстро изменить текущий статус дефекта на предложенный системой (в соответствии со следующим этапом в цикле разработки);
2. Перейти на страницу редактирования проекта путем нажатия на кнопку «Edit» напротив названия проекта, содержащего выбранный дефект;
3. Просмотреть информацию о проекте выбранного дефекта;
4. Добавить в проект выбранного дефекта новый отслеживаемый дефект (см. Рисунок 2.2.5);
5. Просмотреть описание дефекта, текущий статус дефекта, важность дефекта и его логи;
6. Оставить комментарий к дефекту.

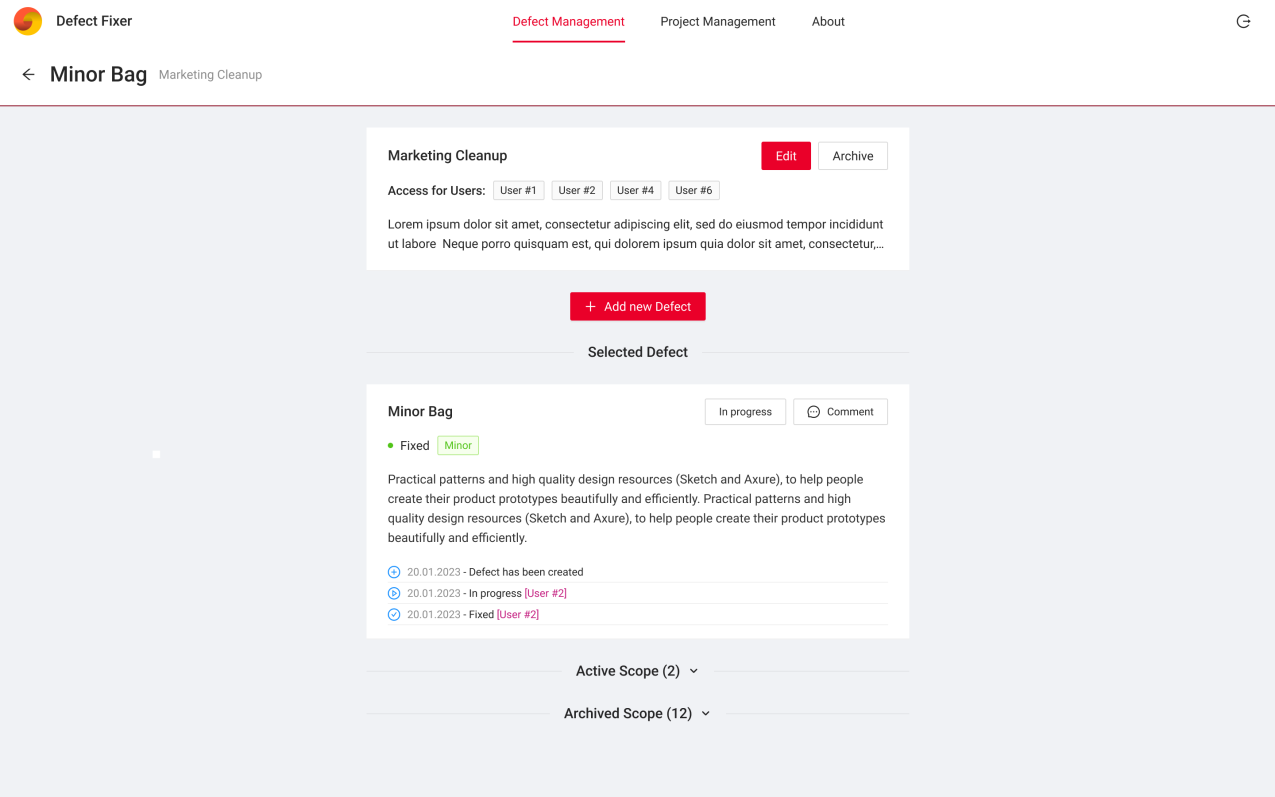


Рисунок 2.2.4 – Страница просмотра дефекта ПП

На этой же странице пользователь системы может раскрыть списки «Active Scope» и «Archived Scope», которые содержат в себе активные и архивированные дефекты соответственно.

После раскрытия одного из списков элемент выбранного на предыдущей стадии дефекта пропадает из ГПИ и он помещается в список блоков с информацией об активных дефектах. Таким образом реализовано разделение списков по категориям.

Если при раскрытии списков все содержащиеся в них дефекты не помещаются на экране ПК, появляется возможность просмотреть их путем вертикального пролистывания страницы.

Страница добавление дефекта может быть вызвана с любой страницы имеющей список дефектов. Сама страница содержит поля для заполнения текстовой информации о дефекте и два ряда опций (то есть можно выбрать только одну одновременно) в виде кнопок, с помощью которых можно назначить важность и приоритет создаваемому дефекту. Создание может быть завершено путем щелчка по кнопке «Create Defect», при условии, что все обязательные поля заполнены. Если условие не соблюдено, пустое поле будет выделено красным цветом. Необязательными являются поля: ожидаемый результат, описание дефекта.

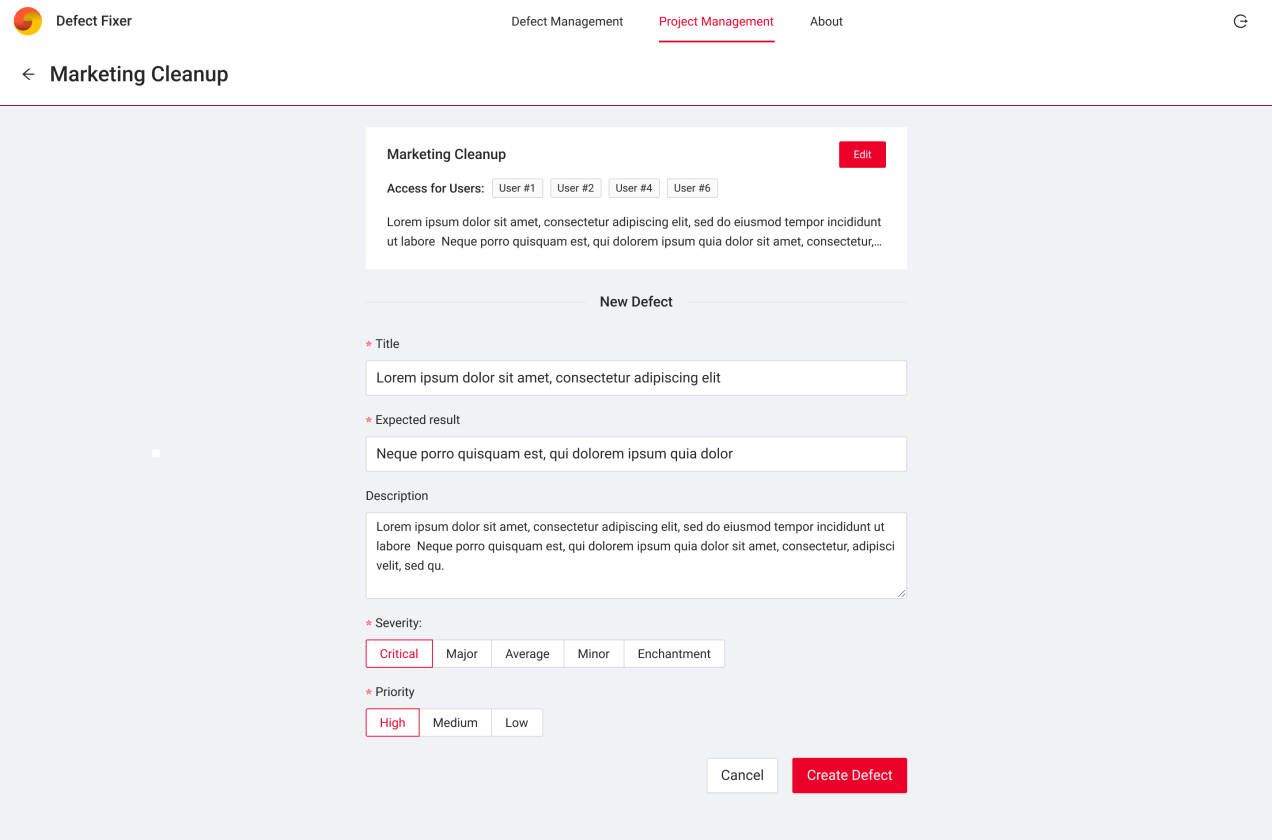


Рисунок 2.2.5 – Страница добавления дефекта ПП в проект

Страница добавления проекта содержит поля, в которые необходимо внести название описание проекта и пользователей которые в дальнейшем будут иметь доступ к дефектам этого ПП. Если проект создает рядовой пользователь его автоматически добавляет к последним. Привилегированный пользователь имеет доступ ко всем проектам и их дефекта по умолчанию.

Страницу добавления проекта полностью повторяет страница редактирования проекта, за исключением того, что текстовые поля заполнены данным о ПП, которые могут быть изменены пользователем (см. Рисунок 2.2.6).

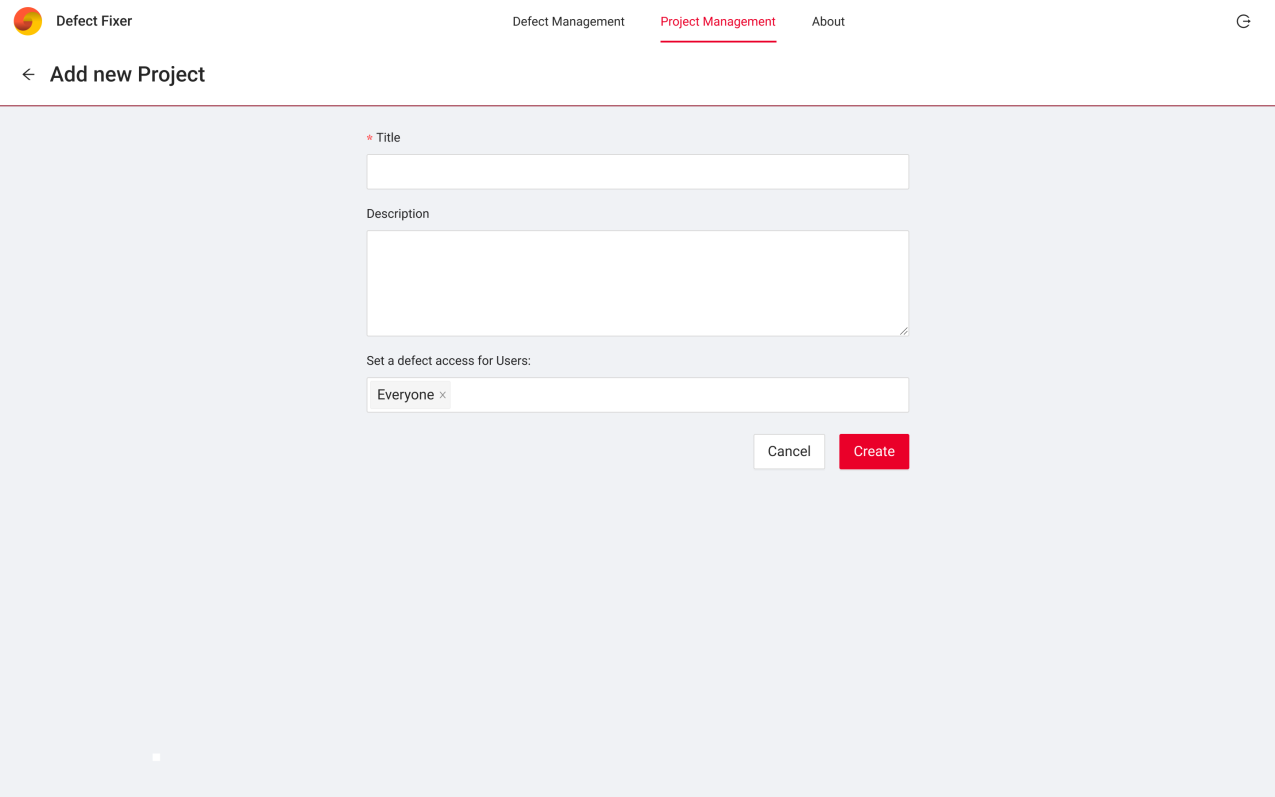


Рисунок 2.2.6 – Страница добавления проекта

Страница управления проектами достижима с любой страницы путем выбора пункта меню «Project Management» в верхней части окна приложения и содержит:

1. Список названий всех доступных пользователю проектов (в случае привилегированного пользователя — всех проектов), при нажатии на каждое из которых пользователь попадает на страницу просмотра дефектов ПП;
2. Общий прогресс исправления дефектов проекта;
3. Список пользователей системы, способных взаимодействовать с определенным проектом;
4. Дату и время создания проекта;
5. Кнопку добавления нового проекта, открывающую страницу добавления проекта;
6. Строку поиска проекта по названию, находящуюся в головке таблицы;
7. Переключающий элемент ГПИ который меняет категорию отображаемых проектов с активных на заархивированные и обратно.

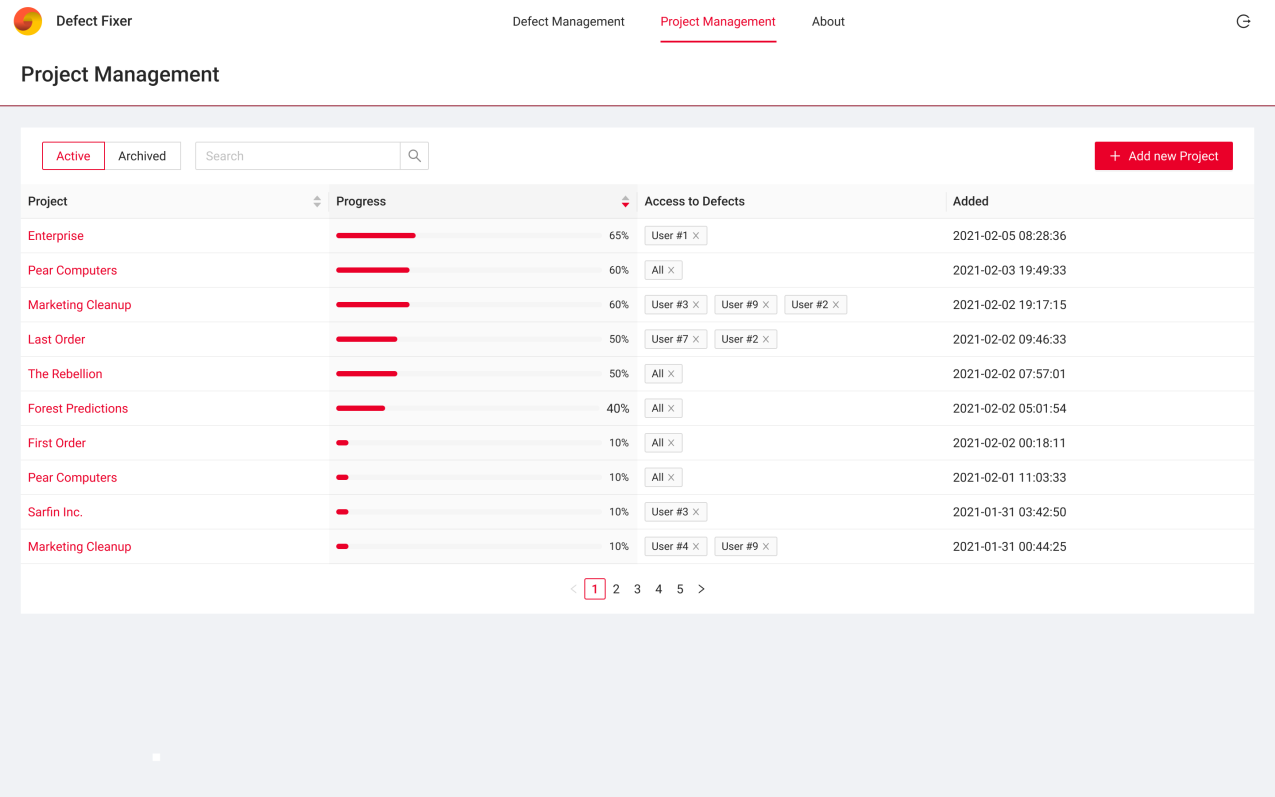


Рисунок 2.2.7 – Страница управления проектами

Таблицы на главной странице приложения и странице управления проектами имеют одинаковый функционал, отвечающий за пролистывание страниц таблицы. Разница между пользовательскими ролями в отношении сценариев информационного взаимодействия заключается в том, что от рядового пользователя в таблицах скрыты дефекты и проекты, к которым у него нет доступа.

На любом из блоков дефекта в списке дефектов присутствует кнопка добавления комментария к дефекту (см. Рисунок 2.2.8), при нажатии на нее появляется всплывающее окно с текстовым полем и кнопками подтверждения и отмены, которое позволяет оставить комментарий к дефекту. После того как пользователь оставит комментарий, он будет отображаться в хронологически отсортированном перечне логов дефекта, в нижней части блока (см. Рисунок 2.2.9).

Все окна пользовательской части системы выполнены в едином стиле и соблюдают все вышеперечисленные ЭТ. Шрифт соответствует стандартам. При отсутствии шрифта в системе и возможности автоматически загрузить его со сторонних источников будет использован один из доступных системных шрифтов.

В ходе разработки эскизов информационных моделей были учтены возможные различия в экранах разных устройств и спроектированы альтернативные виды для более мелких разрешений.

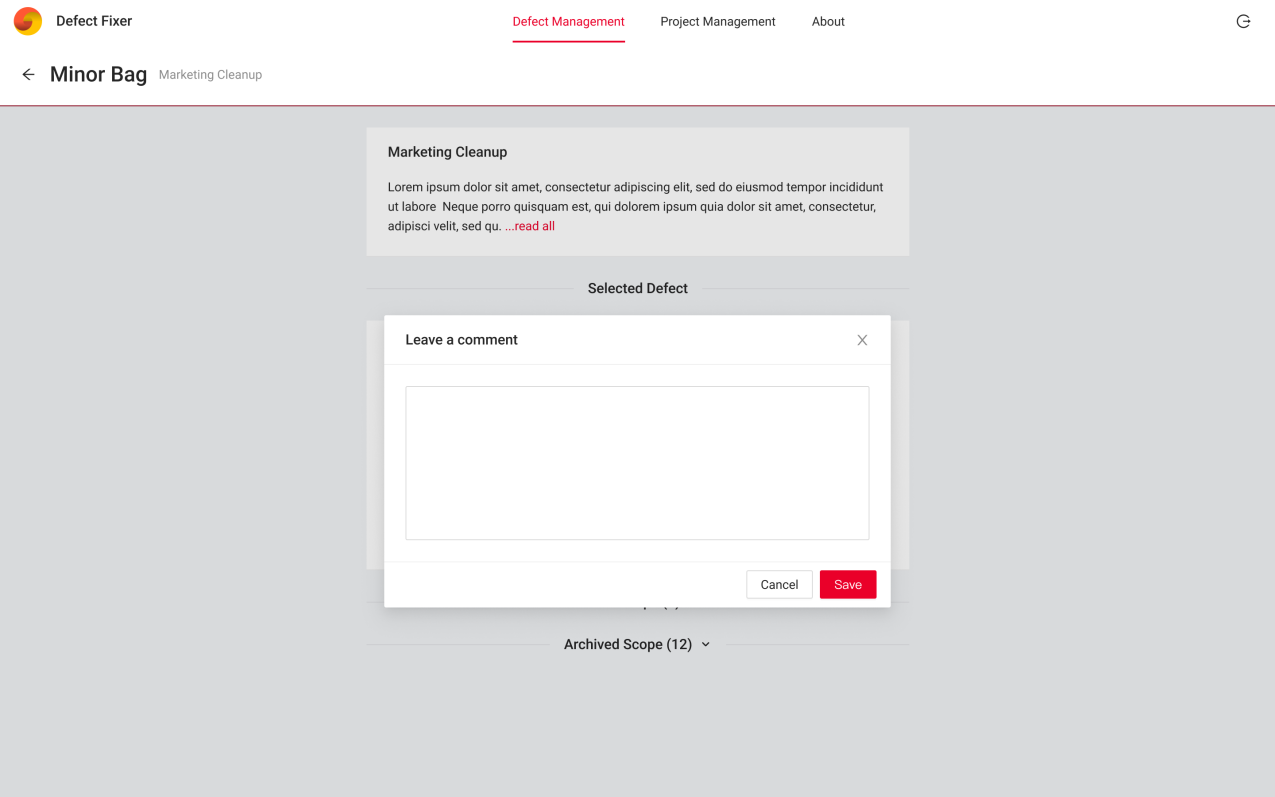


Рисунок 2.2.8 – Окно для добавления комментария

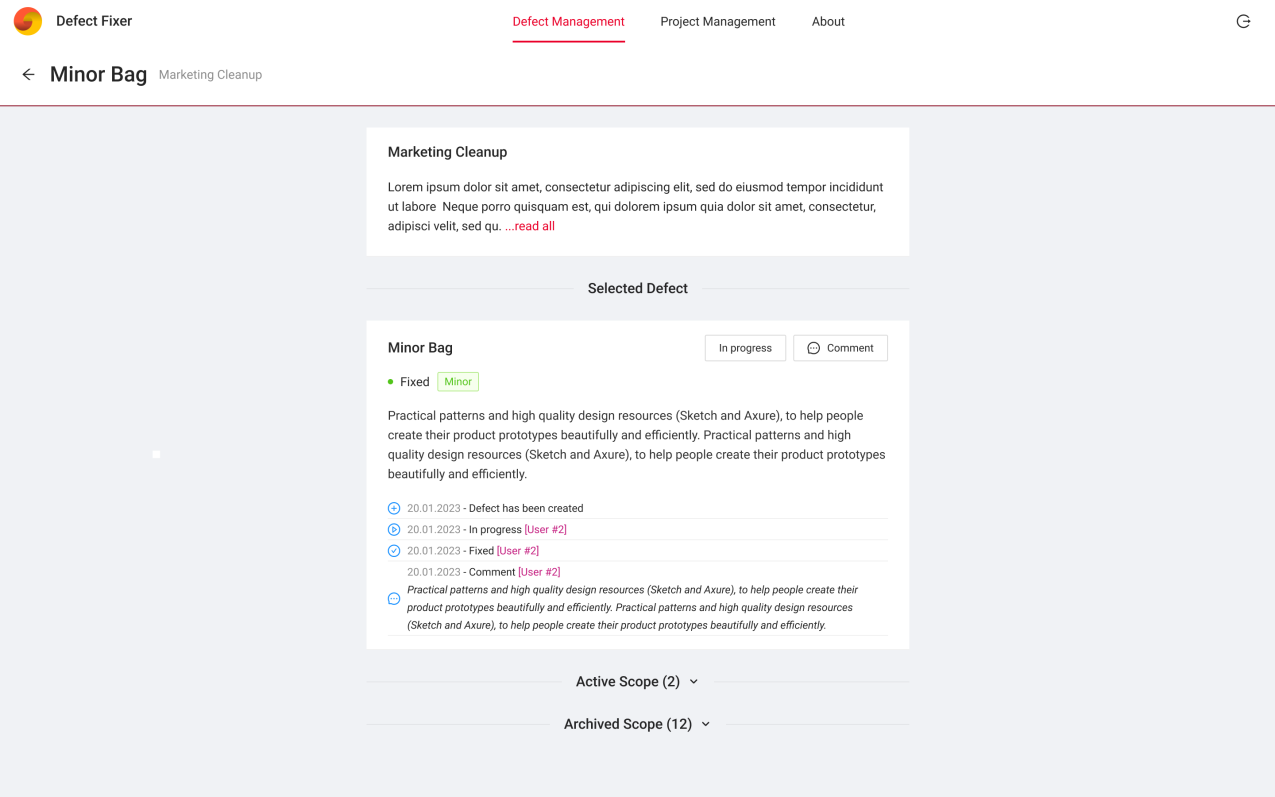


Рисунок 2.2.9 – Пример дефекта с комментарием

2.3 Эргономическая оценка проектируемой системы и выводы

Эргономическая оценка представляет собой комплекс системных методов и научно-технических процедур, направленных на анализ соблюдения эргономических стандартов и норм в проектной документации и образцах средств человеко-машинного взаимодействия.

Оценка включает в себя анализ соответствия технического задания, нормативно-технических актов и руководящих документов, а также разработку рекомендаций для устранения недостатков в соответствии с установленными эргономическими требованиями. Процедура оценки проводится на каждом этапе опытно-конструкторской разработки с целью убедиться в выполнении эргономических норм и определить уровень качества объекта оценки в контексте эргономических параметров.

Таблица 2.3.1 – Общие эргономические требования к проектируемой системе и соответствующие им единичные эргономические показатели

|  |  |  |
| --- | --- | --- |
| Группа ЭТ | Эргономическое требование | Единичный эргономический показатель |
| 1 | 2 | 3 |
| Эргономическое свойство «Управляемость» | | |
| Психологический | П – 1. Один и тот же характер действий на протяжении всего периода работы в системе в схожих ситуациях | Постоянство характера действий |
| П – 2. Наличие указаний на проблемы, возникающие в процессе управления системы | Индикация проблем управления |
| П – 3. Соответствие цветов знаков стереотипам восприятия | Цвет знаков |
| П – 4. Отсутствие неоднозначного толкования требований, инструкций и команд | Однозначность требований |

Продолжение таблицы 2.3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психологический | П – 5. Соответствие количества одновременно предъявляемых сигналов возможностям внимания человека | Количества одновременных сигналов менее четырех |
| П – 6. Наличие индикации степени выполнения операций | Наличие индикаторов степени выполнения операций |
| П – 7. Минимизация сложности и количества шагов для выполнения основных операций управления | Количество шагов для выполнения операций управления |
| П – 8. Обеспечение заметных отличий между органами управления, отвечающими за разные функции | Время нахождения активных элементов ГПИ |
| Психофизиологический | ПФ – 1. Соответствие размеров знаков на экране дисплея оперативному порогу зрения человека | Размер знаков |
| ПФ – 2. Соответствие контраста знаков и фона оптимальным условиям восприятия | Контраст знаков и фона |

Продолжение таблицы 2.3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Психофизиологический | ПФ – 3. Соответствие вида контраста знаков и фона уровню освещенности экрана | Вид контраста знаков и фона |
| ПФ – 4. Яркость и знаков оптимального считывания | Заметность элементов управления |
| ПФ – 5. Отсутствие требований, связанных с управляемостью системы, несоответствующих возможностям органов чувств человека | Видимость элементов управления |
| ПФ – 6. Обозначение недоступных или неактивных элементов управления ГПИ хорошо различимым блеклым цветом | Контраст между основным цветом и цветом неактивных элементов |
| ПФ – 7. Использование пролистываемых и  раскрывающихся списков в целях оптимизации экранного пространства | Наличие раскрывающихся списков в ГПИ |
| Физиологический | Ф – 1. Отсутствие требований, связанных с управляемостью системы, несоответствующих скоростным возможностям человека | Достаточность времени экспозиции элементов |

Продолжение таблицы 2.3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Эргономическое свойство «Освояемость» | | |
| Психологический | П – 9. Соответствие сложности инструкций, предоставляемых обратной связью, времени, отводимому на их усвоение | «Необходимая скорость выполнения» или «Сложность инструкций» |
| П – 10. Исключение открытия ненужных окон | Исключение открытия ненужных окон |
| П – 11. Предоставление навигационных обзоров | Наличие навигационных обзоров |
| П – 12. Однотипное расположение схожих объектов на разных страницах | Распределение объектов на страницах |
| П – 13. Все функциональные элементы интерфейса подписаны или имеют знаки, соответствующие их назначению | Наличие надписей на функциональных элементах ГПИ |
| Психофизиологический | ПФ – 8. Соответствие шрифта стандарту, представленному в ГОСТ ЕН 894-2-2002 [12, с. 5] | Характеристики  шрифта |

Произведем оценку значений индивидуальных эргономических показателей. Рекомендации по определению стандартных значений индивидуальных эргономических показателей формируются на основе нормативно-технической документации и справочных материалов, касающихся эргономики.

Единичные эргономические показатели оцениваются по бинарной шкале, они принимают значение, равное 1, если фактическое значение показателя соответствует рекомендуемому, и равное 0, если оно ему не соответствует. Групповой эргономический показатель рассчитывается как общая оценка по группе единичных показателей:

,

где ∑1 – суммарное число случаев, когда единичный показатель соответствует эргономическому требованию;

∑0 – суммарное число случаев, когда соответствия нет.

Результаты оценки значений единичных и групповых эргономических показателей визуализированы в таблице 2.3.2.

Таблица 2.3.2 – Значения единичных и групповых эргономических показателей проектируемой системы

|  |  |  |  |
| --- | --- | --- | --- |
| Группа ЭТ | Единичные ЭП  с нулевым значением | Значение групповых ЭП | Значение весового коэффициента |
| 1 | 2 | 3 | 4 |
| Эргономическое свойство «Управляемость» | | | |
| Психологический | П – 8. | 8 / (8 + 1) = 0,88 | 0,5 |
| Физиологические | — | 1 / (1 + 0) = 1 | 0,2 |
| Психофизиологический | ПФ – 3. | 7 / (7 + 1) = 0,87 | 0,3 |
| Эргономическое свойство «Освояемость» | | | |
| Психологический | — | 5 / (5 + 0) = 1 | 0,4 |
| Психофизиологические | — | 1 / (1 + 0) = 1 | 0,6 |

С учетом таблиц 2.3.1 и 2.3.2 определим количественное значение эргономических свойств системы:

Присвоим весовые коэффициенты эргономических свойств и вычислим общую эргономичность системы:

; ;

Оценив полученное значение показателя эргономичности, определим ПП как соответствующий заданным базовым значениям характеристик эргономичности.

3 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Структура системы

Исходя из поставленных задач в подразделе 1.3, было решено использовать для разработки системы следующий стек технологий:

1. Electron.js — фреймворк (набор интегрированных библиотек) для создания кросс-платформенных настольных приложений с использованием веб-технологий;
2. JavaScript ES2022 — высокоуровневый, интерпретируемый, нестрого типизированный язык программирования;
3. SASS — препроцессор расширяющий возможности каскадных таблиц стилей (далее — CSS);
4. React.js — библиотека JavaScript для создания пользовательских интерфейсов, которая позволяет создавать компоненты с динамическим и декларативным подходом к отображению данных;
5. MongoDB — это документо-ориентированная NoSQL база данных;
6. Mongoose — библиотека для работы с MongoDB в среде Node.js, предоставляющая средства для определения схемы данных, валидации, создания моделей и выполнения запросов к базе данных с использованием высокоуровневого API;
7. Node.js — среда выполнения JavaScript, позволяющая создавать приложения с использованием асинхронного и событийно-ориентированного подхода;
8. Express.js — это фреймворк для «Node.js», который реализовывает слой функций, необходимых для создания эффективных приложений и API.

Для декомпозиции задач разработки, на модули, была сформирована структурная схема программной части системы, отвечающих поставленным задачам для работоспособности программного обеспечения (см. Рисунок 3.1.1).

Далее в тексте будет использовано сокращение API (Application Programming Interface) — интерфейс программирования приложения, позволяющий подсистемам приложения или его компонентам использовать функционал и данные друг друга, скрыв детали их реализации и предоставление интерфейса для взаимодействия с объектами или сервисами.

Клиентская часть системы написана на основе фреймворка «Electron.js». Приложение, написанное на его основе, включает в себя два процесса при своей работе:

* Главный процесс, управляющий жизненным циклом приложения, включая создание окон и обработку системных событий, последнее достигается путем взаимодействия с API ОС ПК;
* Процесс визуализации (рендеринга), отвечающий за отображение пользовательского интерфейса в окнах приложения, взаимодействующего с Chromium. Последнее позволяет получить доступ к технологиям веб разработки: HTML, CSS, JavaScript и так далее. Он изолирован от главного процесса в целях безопасности и увеличения стабильности.

Межпроцессное взаимодействие (Inter-Process Communication, IPC) в Electron реализуется с помощью механизма IPC. Это позволяет главному процессу и процессу рендеринга обмениваться данными, например, это может быть передача данных из главного во второй для выполнения системных операций (управление окнами приложения, работа с файловой системой, управление устройствами и так далее).

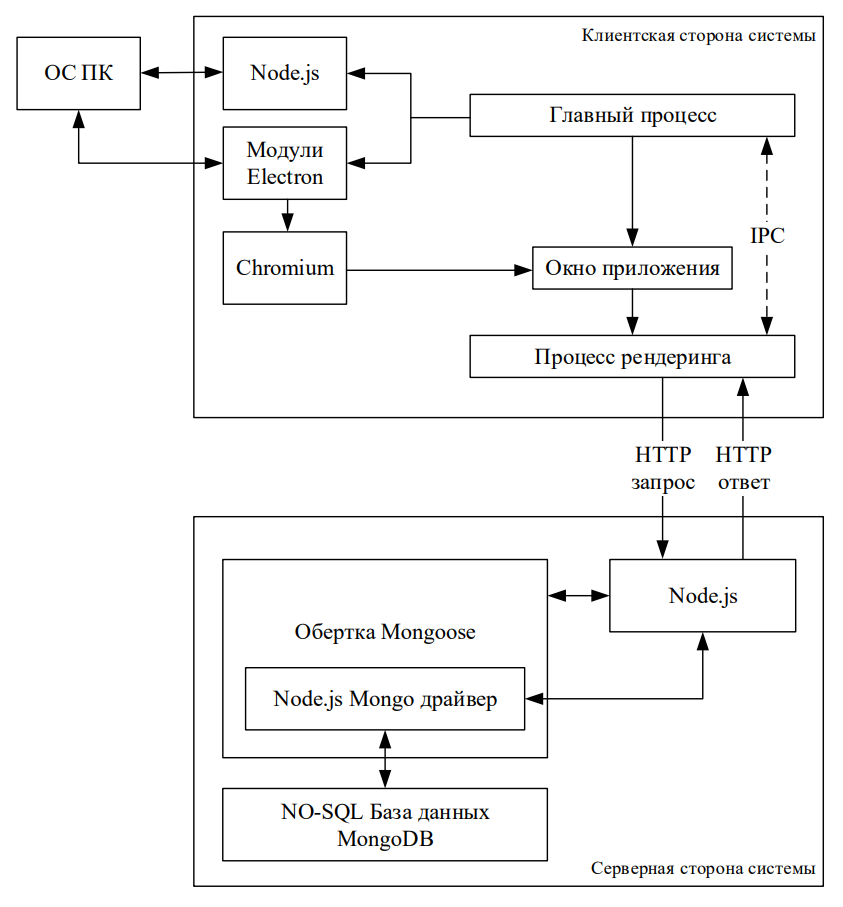


Рисунок 3.1.1 – Структурная схема программной части СОДПП

На схеме приложение, основанное на Electron обозначено как клиентская сторона системы.

Для установки связи между компонентами системы используются HTTP REST API запросы. REST (Representational State Transfer) — это стиль архитектуры программного обеспечения, который определяет набор ограничений для создания распределенных систем. Соблюдение принципов REST предоставляет доступ к ресурсам через стандартные методы HTTP (GET, POST, PUT, DELETE), используя уникальные URI (Uniform Resource Identifier — унифицированный идентификатор ресурса) для каждого ресурса. Построение взаимодействия в данном стиле обеспечивает простоту взаимодействия между подсистемами и обеспечивает использование всех преимуществ унификации, включая повышение согласованности, упрощение использования, улучшение надежности, повышение производительности, а также поддержку масштабируемости.

В данном случае запросы отправляются только из процесса рендеринга клиентского стороны системы, хотя при необходимости возможна реализация и отправки из главного процесса.

Серверная сторона, написанная с использованием «Express.js», принимает данные запросы и обрабатывает их, используя методы модели в Mongoose. Mongoose предоставляет возможность создания схем и моделей данных для работы с MongoDB в среде Node.js. Это позволяет определить структуру данных, их типы, ограничения и взаимосвязи, а также выполнить различные операции с базой данных, такие как создание, чтение, обновление и удаление данных, с использованием стандартизированного API. В ходе разработки было реализован шаблон MVC. MVC (Model-View-Controller) — шаблон (схема) разделения данных приложения и управляющей логики на три отдельных компонента, модификация каждого из которых может осуществляться независимо [13]:

1. Отвечает за представление данных и бизнес-логику приложения. Модель содержит типы данных и методы для их обработки, валидации для сохранения в БД.
2. Представление в данном случае подразумевает под собой преобразование полученной из документов БД информации в необходимый стандартизированный вид.
3. Контроллер принимает запросы от пользователя, обращается к соответствующей модели для получения или изменения данных, а затем обновляет представление, чтобы отобразить изменения пользователю. То есть реализует функции, способные модифицировать информацию в документах БД и налаживать связи между их объектами.

Программная платформа «Node.js» используется на обеих сторонах системы ввиду того она обеспечивает выполнение JavaScript как на сервере так и в приложении для ПК. Это обеспечивает единое окружение выполнения и упрощает разработку приложения, так как один и тот же язык программирования может использоваться для обеих сторон системы и обмена данными между ними.

Сама клиентская часть системы написана на «React.js» и стилизована с помощью SCSS. «React.js» позволяет использовать компонентный подход при написании кода, это позволяет инкапсулировать необходимую функциональность в компонентах и повторно использовать их, что может упростить соблюдение принципа сокращения дублирования кода в ПО (принцип DRY). Так же библиотека увеличивает производительность приложения путем частичного, а не полного обновления страниц ПП. Использование SCSS в свою очередь упрощает восприятие таблиц стилей, так как допускает использование их вложенности, позволяет использовать функции внутри CSS и разбивать по файлам, что совершенствует структуру файлов при написании программного кода.

Все из перечисленных методик и технологий широко используются в индустриальной практике, что позволяет расширить число потенциальных свободных разработчиков, способных модифицировать проект. В ходе разработки были сформированы API и написаны структуры, призванные продемонстрировать возможность модернизации и наметить области функциональности наиболее перспективные для дальнейшей разработки.

3.2 Реализация клиентской стороны

На сегодняшний момент большинство систем, имеющих клиент–серверную архитектуру, и наша система в частности, перекладывает реализацию основного функционал на сторону пользователя. Связанно это с увеличением объемов обрабатываемых данных, общим обогащением видов ГПИ, платформенной адаптивностью и распространенностью одностраничных приложений. Последние вызвано растущими ожиданиями пользователей от интерактивности приложений, а также с потребностью создания многофункциональных интерфейсов, способных эффективно обрабатывать большие объемы данных [14]. Приложение, являющееся клиентской стороной СОДПП, использует архитектуру одностраничного приложения и написана на «React.js», что как говорилось ранее позволяет построить ГПИ на основе компонентного подхода.

В случае компонентного подхода приложение строится на основе блоков (компонентах). Один компонент содержится в одном файле формата JSX и содержит в себе полную реализацию. Иногда часть логики выносится в отдельный блок как показано на рисунках 3.2.1 — 3.2.2.

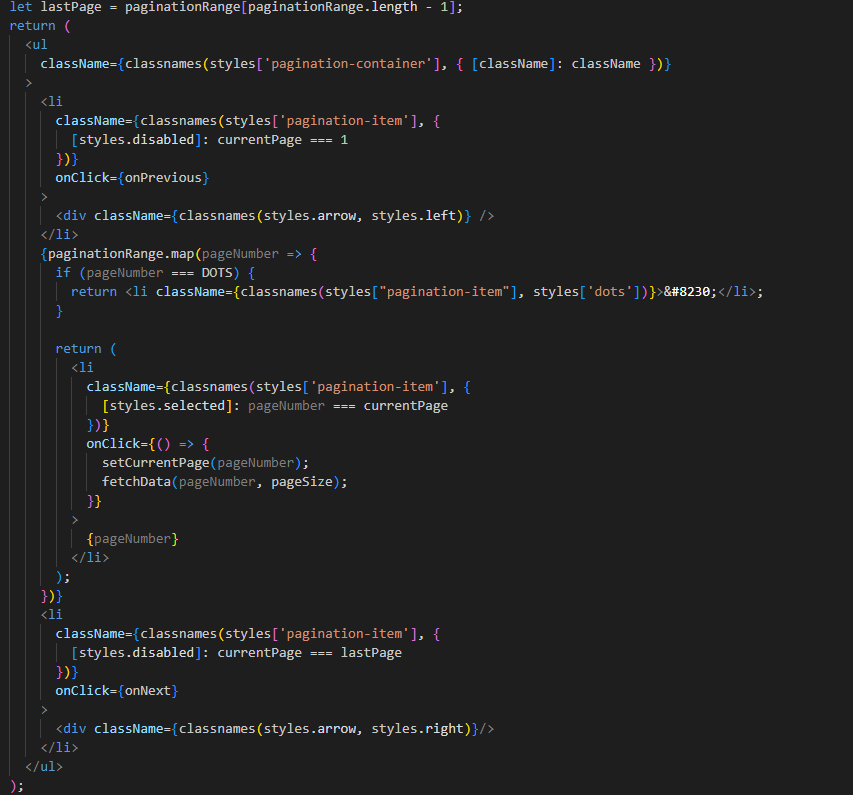


Рисунок 3.2.1 – Реализация компонента переключение страниц таблицы

При разработке использована стилизация с помощью модулей, то есть стили импортируются в компонент как объект со списком полей, а при финальной сборке проекта каждому из стилей присваивается уникальный идентификатор компонента.

В основной своей части ПО клиентской стороны написано, используя функциональный подход. Управление состояниями компонентов в виртуальном дереве элементов осуществляется через «хуки». Данный подход рекомендуются самими разработчиками фреймворка, при условии написания новых приложений. «Хук» — специальная функция, призванная управлять состоянием компонента, что и делает его реактивным [15]. В целом, «хуки» предоставляют более простой и чистый синтаксис по сравнению с классами, упрощая разработку и поддержку кода.

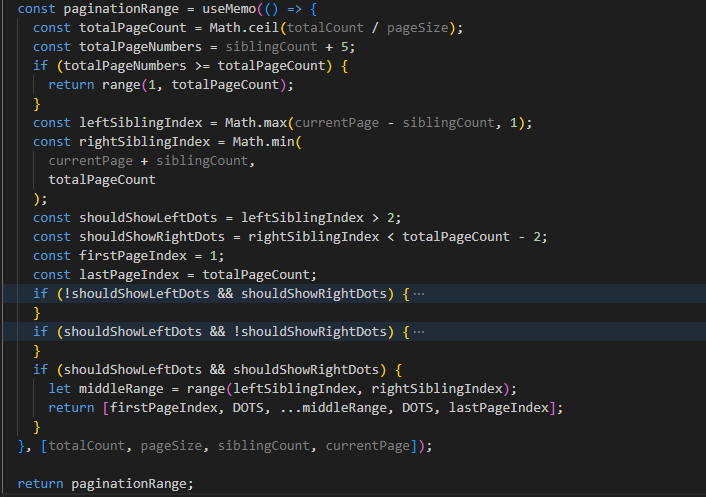


Рисунок 3.2.2 – Реализация логики переключения страниц таблицы

Реализация приложения для ПК на выбранном языке программирования, значительно отличается архитектурно от разработки такого на других технологических стеках — «React.js» для работы необходима среда выполнения, на это способен движок внутри любого браузера, или в какой либо другой среде выполнения. В данном случае движок V8 используется в Chromium, которые в свою очередь запускается в процессе «Node.js» и использует его как окружение. Главной особенностью приложений, построенных на таким образомзаключается в архитектурном разделении программы на два основных процесса.

Концепция множественных процессов наследуется от браузеров, использующих в себе Chromium. Помимо отображения содержимого окна приложения должно выполнятся множество второстепенных функций, таких как управление множеством окон (или вкладок) и загрузка сторонних расширений. Организация такой архитектуры:

* позволит повысить устойчивость и производительность приложения;
* делает возможным распределение нагрузки между изолированными друг от друга процессами;
* обеспечит изоляцию в случае сбоев системы или целенаправленного стороннего вмешательства.

Кроме этого разделение на основной процесс и процессы рендеринга делает возможным модульную структуру приложения, что снижает сложность восприятия кода и улучшает его поддерживаемость.

Поскольку основной процесс и процесс рендеринга имеют разные обязанности в модели таких процессов, IPC является единственным способом выполнения многих общих задач, таких как вызов собственного API из пользовательского интерфейса или запуск изменений в содержимом из меню [16]. При разработке приложений на основе «Electron.js» реализация IPC становится одним из основных компонентов для успешной реализации ПП.

В разрабатываемом приложении разработан так называемый контекстный мост, налаживающий связи между двумя процессам приложения (см. Рисунок 3.2.3).

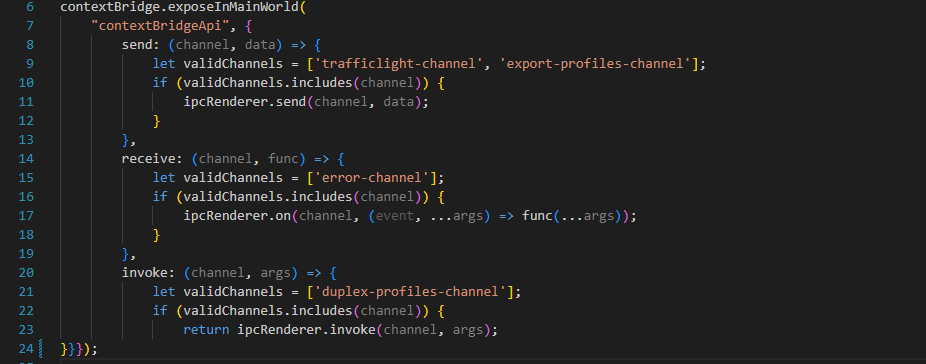


Рисунок 3.2.3 – Программная реализация контекстного моста

Первые два метода объекта позволяют отправлять данные в одностороннем порядке, по ранее предусмотренным каналам. Такое поведение является необходимым и достаточным при отсутствии нужды в установлении обратного канала связи.

На рисунке 3.2.4 приведена реализация примера использования дуплексной связи главного процесса и процессов рендеринга.

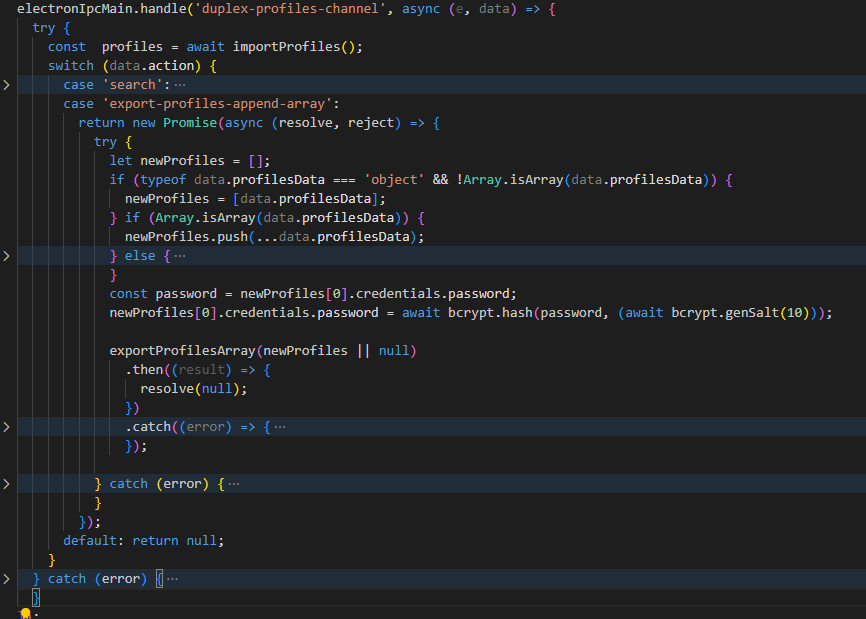


Рисунок 3.2.4 – Пример реализации дуплексной связи процессов

Запросы на сервер, как правило, шлются из компонентов, или страниц, на которых они находятся, если необходимо отобразить данные сразу после загрузки страницы.

Остальной программный код, необходимый для функционирования системы, предоставлен в приложении А данной пояснительной записки.

3.3 Структура базы данных информационной системы

В данном разделе предоставлен обширный анализ внутренней структуры БД разрабатываемой системы. При разработке системы была использована система управления БД «Mongo». «MongoDB» относится к классу NoSQL БД, они разработаны для хранения и управления разнородными типами данных, используя документо-ориентированный подход к хранению информации [17]. «MongoDB» предоставляет гибкую схему данных, масштабируемость и производительность, что и является ее преимуществами.

В ходе анализа БД следует учесть, что базы, построенные по принципу No-SQL сильно отличаются от табличных и имеют свою специфику [17], а именно:

* подход моделирования принципиально отличается от SQL БД, при формировании структуры БД в основном учитывается форма ответов на запросы;
* сама БД состоит не из таблиц, а из коллекций JSON–подобных документов, которые содержат в себе поля с данными;
* документы такой БД не требует строгой типизации, а их состав формируется исходя из требований системы, путем написания моделей документов необходимой строгости типизации.

Ввиду данной специфики No-SQL схемы БД подобного типа несколько отличаются от табличных. Физическая модель БД СОДПП приведена на рисунке 3.3.1.

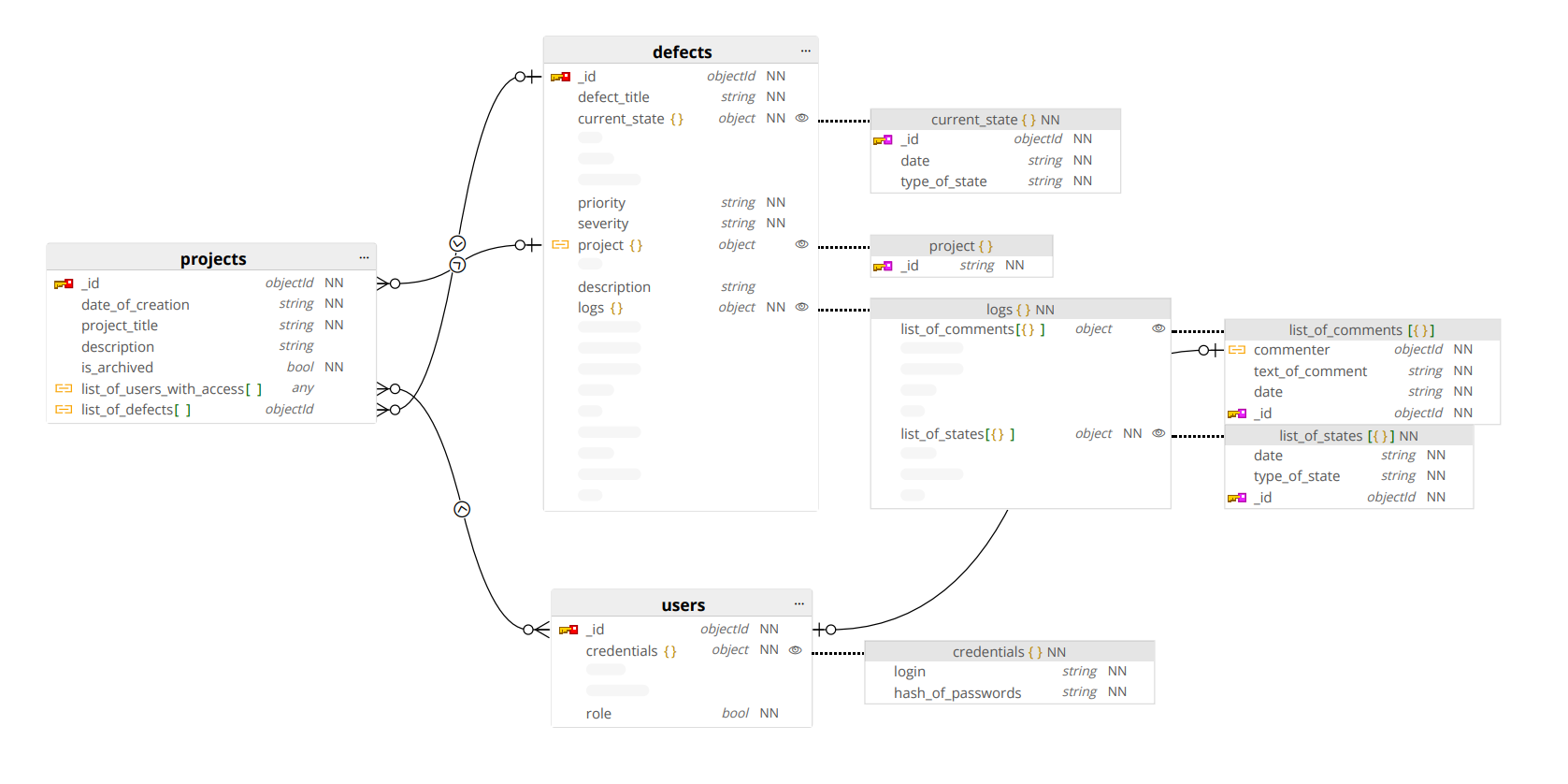


Рисунок 3.3.1 — Физическая модель БД СОДПП

Представленное на рисунке 3.3.1 обозначение NN (ненулевое значение) означает то что поле документа будет заполненным на стадии добавления в БД, если поле является пустым в самом запросе о добавлении, функционал, описанный в модели объекта заполнит поле данными вычисляемыми по умолчанию. Некоторые поля вынесены из блока модели самого документа вследствие того, что поле является отдельным типизированным объектом или массивом типизированных объектов. Ввиду особенностей структуры документов БД было решено использовать в случае, если поле является подобъектом его название до символа точки, а наименование его полей после него.

Сформируем таблицы структур документов данной БД. Список полей модели «Пользователь» представлена в таблице 3.3.1.

Таблица 3.3.1 – Структура модели документа «Пользователь»

|  |  |  |
| --- | --- | --- |
| Наименование поля документа | Тип | Примечание |
| \_id | ObjectId | Идентификатор аккаунта |
| credentials.login | String | Логин аккаунта пользователя |
| credentials. hash\_of\_passwords | String | Личный хэш пароля пользователя |
| role | Boolean | Текущая роль пользователя |

Хэш пароля пользователя представляет собой результат работы криптографической функции, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины, выполняемое определённым алгоритмом [18].

Структура полей модели документа «Дефекты» представлена в таблице 3.3.2.

Таблица 3.3.2 – Структура модели документа «Дефекты»

|  |  |  |
| --- | --- | --- |
| Наименование поля документа | Тип | Примечание |
| \_id | ObjectId | Идентификатор дефекта |
| defect\_title | String | Название дефетка |
| current\_state | state\_type | Текущее состояние дефекта |
| priority | String enum | Приоритет дефекта |
| severity | String enum | Важность дефекта |
| project.\_id | ObjectId | Идентификатор проекта дефекта |
| description | String | Текстовое описание дефекта |
| logs | Object | Логи дефекта |

Enum в данном случае обозначает, что строка может иметь только одно и заранее назначенных значений. Object указывает на то что поле документа может принимать объект, поля которого строго типизированы. Список типизированных полей объекта «Логи» («Logs») представлен в таблице 3.3.3.

Таблица 3.3.3 – Структура модели документа «Логи»

|  |  |  |
| --- | --- | --- |
| Наименование поля документа | Тип | Примечание |
| list\_of\_comments | Массив объектов сomments | Список комментариев дефекта |
| list\_of\_comments | Массив объектов state\_type | Список всех прошлых состояний дефекта |

Документ, структура которого показана в таблице 3.3.3, содержит в себе массивы объектов пользовательского типа, а документ, отображенный в таблице 3.3.2, содержит поле имеющий пользовательский тип «state\_type». Описание полей данного типа предоставлено в таблице 3.3.4 данного раздела пояснительной записки.

Таблица 3.3.4 – Перечень полей пользовательского типа «state\_type»

|  |  |  |
| --- | --- | --- |
| Наименование поля типа | Тип | Описание поля типа |
| date | String | Дата установки статуса дефекта |
| type\_of\_state | String enum | Название статуса дефекта |

В таблице 3.3.3 поле «list\_of\_comments» имеет в качестве типа массив объектов, соответствующих пользовательскому типу «сomments». Описание полей данного типа предоставлено в таблице 3.3.5.

Таблица 3.3.5 – Перечень полей пользовательского типа «comments»

|  |  |  |
| --- | --- | --- |
| Наименование поля типа | Тип | Описание поля типа |
| \_id | ObjectId | Идентификатор комментария к дефекту |
| commenter | ObjectId | Идентификатор пользователя, оставившего комментарий |

Продолжение таблицы 3.3.5

|  |  |  |
| --- | --- | --- |
| Наименование поля типа | Тип | Описание поля типа |
| text\_of\_comment | String | Текст комментария дефекта |
| date | String | Дата и время когда был оставлен комментарий к дефекту |

Структура модели документов типа проектов «projects» представлена в таблице 3.3.6.

Таблица 3.3.6 – Структура модели документа «Проекты»

|  |  |  |
| --- | --- | --- |
| Наименование поля документа | Тип | Примечание |
| \_id | ObjectId | Идентификатор проекта |
| date\_of\_creation | String | Дата создание проекта |
| project\_title | Sting | Название проекта |
| description | String | Текстовое описание проекта |
| is\_archived | Boolean | Флаг, указывающий на то заархивирован проект или нет |
| list\_of\_users\_with\_access | Массив ObjectId | Список идентификаторов пользователей имеющих доступ к проекту |
| list\_of\_defects | Массив ObjectId | Список идентификаторов дефектов проекта |

Вложенность некоторых объектов необходима ввиду некоторых правил написания моделей в Mongoose. Все ObjectId являются не только идентификаторами объектов, но и ссылками на них, по которым, при необходимости возможно получить данные об объекте.

В данном подразделе мы в полной мере описали структуру БД информационной системы, реализованной путем использования No-SQL инструмента MongoDB.

3.4 Выводы и оценка результатов разработки

В разработанной информационной системе реализован весь необходимый для базового программного решения функционал. Для его реализации использовались следующие технологии: «Electron.js», JavaScript, SASS, «React.js», MongoDB, Mongoose, «Node.js», «Express.js».

Так же система предоставляет расширенные возможности при выполнение следующих задач: просмотр учтенных в системе ПП, просмотр дефектов учтенных в системе ПП, добавление новых дефектов ПП в систему и изменение информации об объектах учета, путем добавления дополнительных информационных полей в каждый из перечисленных объектов.

Пользовательская сторона системы реализует интуитивно понятный и простой в использовании ГПИ, основанный на компонентном методе проектирования. Архитектура всей системы построена в соответствии с современными индустриальными практиками и позволяет интегрировать новые программные модули (или изменять уже существующие) без больших временных затрат.

Структурой приложения предусмотрена возможность расширения функционала, путем написания API для присоединения новых программных блоков и проектирования самой системы по популярным на момент разработки паттернам проектирования. Использование последних позволяет стандартизировать интерфейсы доступа к внутреннему наполнению системы, что облегчает масштабирование системы для сторонних разработчиков ПО. Все принципы, задействованные в ходе разработки, являются популярными практиками в сообществе разработки, что может способствовать стимулированию экосистемы вокруг приложения, позволяя разработчикам создавать дополнительные инструменты, расширения и модификации.

Безопасность и конфиденциальность информации находятся на достаточном для данной системы уровне, что обеспечено использованием ранее проверенным шаблонам проектирования. Такой подход позволяет реализовать строгий контроль доступа к данным и внутренним функциям приложения, что и обеспечивает необходимую защиту системы от сторонних вмешательств.

4 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В рамках функционального тестирования приложение исследовано на соответствие требованиям, сформулированным ранее.

Тестированию в основном подвергает пользовательская сторона системы — дефекты серверной стороны проявляются при таком подходе самостоятельно. Ввиду широкого функционала приложения отчет о тестировании содержит общие, более широкие тестовые примеры.

В процессе тестирования, исходя из поставленной задачи, были принято решение использовать композицию тестов в виде наборов функционального тестирования, а так же тестирование ГПИ при помощи одного из экспертных подходов — эвристик Якоба Нильсена.

Следует пояснить, что цикл исправления дефекта, упомянутый далее по тексту, имеет следующий вид: «открытый (Open) — в процессе исправления (In progress) — исправленный (Fixed) — архивированный или снова открытый». То есть, после того как дефект получает статус исправленного, кто-то, из имеющих доступ к дефектам данного ПП, может либо окончательно скрыть дефект из списка активных, либо заново присвоить ему статус отрытого, указав на то, что дефект не был окончательно исправлен.

Для функционального тестирования приложения были разработаны и использованы наборы тестовых проверок (далее — «тест–кейсов») и внесены в таблицу 4.1 [19].

По всем найденным в процессе тестирования дефектам СОДПП необходимо составить отчет и представить его в виде таблицы (см. Таблица 4.2).

Таблица 4.1 – Текст-кейсы функциональных проверок СОДПП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 1 Приложение позволяет успешно зарегистрироваться | Регистрация | 1 Запустить приложение  2 Щелкнуть на кнопку ГПИ с надписью «Sign Up»  3 В поле «Login» ввести «login\_example»  4 Ввести в поле «Enter new password» значение «Qw1Testtest»  5 Ввести в поле «Confirm new password» значение «123»  6 Повторно щелкнуть на кнопку ГИП с надписью «Sign Up» | 1 Открывается экран регистрации  2 Поле заполняется  3 Поле заполняется  4 Поле заполняется  4 Пользователь переходит на главную страницу | Успех |
| 2 Приложение позволяет успешно авторизоваться | Авторизация | 1 Запустить приложение  2 Щелкнуть на кнопку ГПИ с надписью «Sign In»  3 В поле «Login» ввести «login\_example»  4 Ввести в поле «Password» значение «C0rrect\_passWord»  5 Щелкнуть на кнопку ГПИ с надписью «Sign In» повторно | 1 Открывается экран авторизации  2 Поле заполняется  3 Поле заполняется  4 Пользователь переходит на главную страницу | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 3 При попытке регистрации аккаунта с невалидными данными отображается сообщение об ошибке | Регистрация | 1 Запустить приложение  2 Щелкнуть на кнопку ГПИ с надписью «Sign Up»  3 Оставить поле «Login» пустым  4 Ввести в поле «Enter new password» значение «123»  5 Ввести в поле «Confirm new password» значение «123» | 1 Открывается экран регистрации  2 Необходимое поле заполняется  3 Отображается всплывающее сообщение «Such profile doesn't exist» | Успех |
| 4 При попытке входа с использованием не регистрированных ранее реквизитов для входа отображается индикация ошибки | Авторизация | 1 Запустить приложение  2 Щелкнуть на кнопку ГПИ с надписью «Sign In»  3 В поле «Login» ввести «incorrectlogin»  4 Ввести в поле «Password» значение «incorrect\_password»  5 Щелкнуть на кнопку ГПИ с надписью «Sign In» повторно | 1 Открывается экран авторизации  2 Поле заполняются некорректно  3 Поле заполняются некорректно  4 Поля с некорректным вводом обводятся красным цветом  5 Отображается всплывающее сообщение «Incorrect credentials» | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 5 Отображение таблицы с перечнем всех дефектов, ранее внесенных в СОДПП | Просмотр таблицы дефектов | 1 Запустить приложение как авторизированный пользователь  2 Зайти на главную страницу приложения | 1 Открывается окно таблицы дефектов  2 Выводится перечень в виде таблицы, соответствующей спроектированному во втором разделе | Успех |
| 6 Поиск дефектов по текстовым полям | Просмотр таблицы дефектов | 1 Запустить приложение как авторизированный пользователь  2 Зайти на главную страницу приложения  3 Ввод поискового запроса в соответствующее текстовое поле  4 Щелкнуть по клавише ввода или кнопке с пиктограммой поиска | 1 Поле поиска дефектов по текстовым полям является изменяемым и активным  2 Отображается вводимые в поле поиска символов  3 Текущая таблица заменяется таблицей дефектов, удовлетворяющих поисковому запросу | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 7 Фильтрация дефектов по характеристикам | Просмотр таблицы дефектов | 1 Запустить приложение как авторизированный пользователь  2 Зайти на главную страницу приложения  3 Выбрать одну из характеристик «Priority» и «Severity» в элементе выпадающего списка ГПИ  4 Щелкнуть по кнопке ГПИ с надписью «Apply Filters» | 1 Поля характеристик принимают выбранные пользователем значения  2 Отображаются варианты значений характеристик  3 Текущая таблица заменяется таблицей дефектов, удовлетворяющих поисковому запросу фильтрации | Успех |
| 8 Отображение таблицы с перечнем отслеживаемых ПП | Просмотр таблицы ПП | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра таблицы отслеживаемых ПП | 1 Открывается главная страница приложения  2 Открывается страница с таблицей просмотра ПП, соответствующая спроектированному во втором разделе макету | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 9 Корректное пролистывание таблиц отлеживаемых объектов | Просмотр таблицы ПП.  Просмотр дефектов | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу с таблицей любого из отслеживаемых в системе объектов  3 Щелкнуть по номеру одной из страниц таблицы или кнопке с пиктограммой стрелки в ГПИ | 1 Открывается окно, содержащее таблицу  2 Содержание строк таблицы меняются при ее пролистывании  3 Строки таблицы не повторяются  4 Пролистывание в обе стороны ограничено | Успех |
| 10 Добавление нового ПП в систему | Добавление нового ПП | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра таблицы отслеживаемых ПП  3 Щелкнуть по кнопке с надписью «Add new Project» в ГПИ  4 Ввести название ПП в поле ввода  5 Ввести текстовое описание ПП в поле ввода  6 Ввести логины пользователей имеющих доступ в данному ПП  7 Щелкнуть по кнопке подтверждения ввода в ГПИ | 1 Открывается страница добавления дефектов  2 Данные в полях ввода формы изменяются в зависимости от вводимого пользователем значения  3 После подтверждения страница текущая закрывается  4 Открывается предыдущая страница клиентского приложения | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 11 Добавление нового дефекта в ПП | Добавление дефекта в ПП | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть на кнопку с надписью «Add new Defect» в ГПИ  4 Заполнить текстовые поля формы добавления  5 Выбрать характеристики «Severity» и «Priority» из предложенных вариантов  6 Щелкнуть по кнопке подтверждения добавления дефекта | 1 Отображается страница добавления нового дефекта в ПП  2 Данные в полях ввода формы изменяются в зависимости от вводимого пользователем значения  3 Новый дефект отображается при возврате на страницу списка дефектов ПП | Успех |
| 12 Изменение информации о ПП | Изменение информации о ПП | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть по кнопке с надписью «Edit» возле названия дефекта  4 Изменить необходимую информацию о дефекте  5 Щелкнуть по кнопке подтверждения изменения дефекта | 1 Отображается страница изменения дефекта в ПП  2 Данные в полях ввода формы изменяются в зависимости от вводимого пользователем значения  3 Данные о дефекте на странице списка дефектов ПП изменяются на данные введение в форме после подтверждения ввода | D – 1 |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 13 Изменение информации о дефекте ПП | Изменение дефекта | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть по кнопке с надписью «Edit» возле названия ПП  4 Изменить необходимую информацию о ПП  5 Щелкнуть по кнопке подтверждения изменения ПП в ГПИ | 1 Корректно отображается страница изменения информации дефекта  2 После подтверждения, открывается предыдущая страница приложения  3 Данные изменяются на ранее введенные в форму в соответствующем дефекте | Успех |
| 14 Изменение статуса дефекта ПП | Изменение дефекта | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть по кнопке с надписью соответствующей названию статуса, следующего за текущим в цикле исправления дефектов | 1 Отображается список дефектов и информация о ПП, к которому они относятся  2 Текущий статус меняется  3 Кнопка с переключением статуса изменяется в соответствии с циклом исправления дефекта | D – 2 |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 15 Просмотр списка дефектов ПП | Просмотр информации о ПП | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП путем нажатия на название ПП или дефекта к нему относящегося | 1 Отображается список текущих дефектов  2 Отображается список архивированных дефектов  3 Каждый из блоков содержит актуальную и полную информацию о дефекте  4 Кнопки изменения состояния активны  5 Отображается текущий статус дефекта | Успех |
| 16 Архивирование отслеживаемых объектов | Добавление в архив | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть по кнопке с надпись «Archive» напротив названия дефекта  3 Щелкнуть по кнопке с надпись «Archive» напротив названия ПП | 1 Объекты отслеживания, которые были заархивированы пропадают из текущего вида ГПИ  2 Заархивированные объекты помещаются в соответствующую категорию (раздел) | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 17 Просмотр архива отслеживаемых объектов | Просмотр архива | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра таблицы с перечнем отслеживаемых ПП  3 Переключиться на просмотр архивированных ПП путем нажатия на кнопку «Archived» переключающего элемента ГПИ  4 Щелкнуть по названию любого из архивированных проектов  5 Щелкнуть по элементу ГПИ с надписью «Archived Scope» внизу окна | 1 Открывается страница с соответствующей таблицей  2 Таблица с перечнем ПП отображает ПП нужной категории  3 Содержание списка дефектов соответствует нужной категории  4 Состав информации о дефектах соответствует макету, разработанному во втором разделе | D – 3 |
| 18 Добавление комментария к дефекту ПП | Добавление комментария | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть по кнопке с надписью «Comment» напротив названия любого из дефектов  4 Ввести текст комментария в поле ввода  5 Щелкнуть по кнопке с надписью «Save» внизу всплывающего окна | 1 Открывается страница просмотра списка дефектов ПП  2 Всплывает окно с формой добавления комментария  3 Кнопка подтверждения закрывает сплывающее окно и добавляет комментарий в конец логов дефект | Успех |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование | Модуль | Описание | Ожидаемый результат | Статус |
| 19 Просмотр информации о дефекте ПП | Просмотр информации о дефекте ПП | 1 Запустить приложение как авторизированный пользователь  2 Щелкнуть по названию дефекта | 1 Открывается страница просмотра информации о выбранном ранее дефекте  2 Состав информации соответствует приведенному в макете, разработанном во втором разделе  3 Отображается актуальный статус дефекта  4 Кнопки взаимодействия с дефектов активны  5 Система корректно отображает логи дефекта в хронологическом порядке  6 Блок и информацией о выбранном дефекте закреплён до начала категориальных списков | Успех |
| 20 Полное закрытие программы | Закрытие приложения | 1 Запустить приложение как неавторизированный пользователь  2 Нажать на пиктограмму крестика в правом верхнем углу окна ПП | 1 Открывается экран авторизации  2 Приложение полностью закрывается | Успех |

Таблица 4.2 – Описание найденных дефектов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название дефекта | Важность | Алгоритм воспроизведения | Фактический результат | Ожидаемый результат | Прило-жение |
| D – 1 Измененные данные о дефекте не сохраняются | Высокая | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра списка дефектов ПП  3 Щелкнуть по кнопке с надписью «Edit» возле названия дефекта  4 Изменить необходимую информацию о дефекте  5 Щелкнуть по кнопке подтверждения изменения дефекта | После подтверждения внесения изменений и возврата на прошлую страницу приложения дефект не претерпевает изменений | После подтверждения внесения изменений и возврата на прошлую страницу приложения данные о дефекте изменятся | — |

Продолжение таблицы 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название дефекта | Важность | Алгоритм воспроизведения | Фактический результат | Ожидаемый результат | Прило-жение |
| D – 2 Следующий статус дефекта не соответствует циклу исправления дефектов | Низкая | 1 Запустить приложение как авторизированный пользователь | Следующий статус дефекта не соответствует циклу исправления дефектов | Надпись на кнопе изменения статуса дефекта и статус дефекта являются близстоящими в цикле исправления дефекта | Рис. 4.1 |
| D – 3 Список архивированных дефектов ПП включает в себя не только дефекты, которые были заархивированы | Очень низкая | 1 Запустить приложение как авторизированный пользователь  2 Зайти на страницу просмотра таблицы с перечнем отслеживаемых ПП  4 Щелкнуть по названию проектов в котором есть заархивированные дефекты  5 Щелкнуть по элементу ГПИ с надписью «Archived Scope» внизу окна | Список заархивированных дефектов включает в себя все дефекты ПП | Список заархивированных дефектов включает в себя лишь заархивированные  дефекты | Рис.  4.2 |

На рисунке 4.1 демонстрируется дефекта «Следующий статус дефекта не соответствует циклу исправления дефектов». Текущий статус (под названием дефекта) и следующий возможный к установке (на красной кнопке справа) совпадают, что не соответствует порядку цикла исправления дефекта.

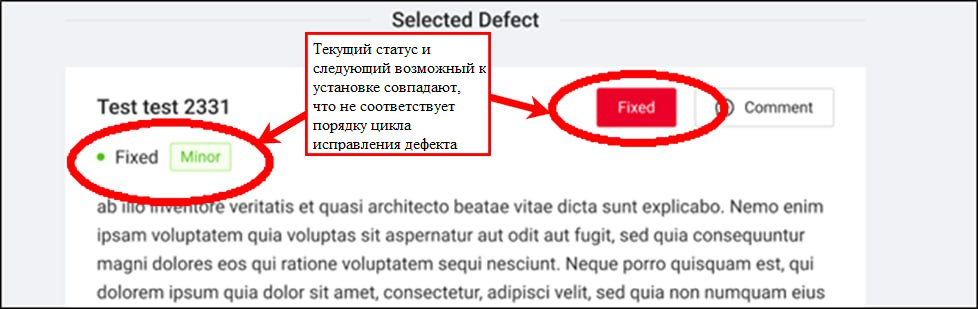


Рисунок 4.1 — Демонстрация дефекта D – 2

На рисунке 4.1 демонстрируется дефекта «Список архивированных дефектов ПП включает в себя не только дефекты, которые были заархивированы». Первый дефект списка не является архивированным и имеет статус открытого.

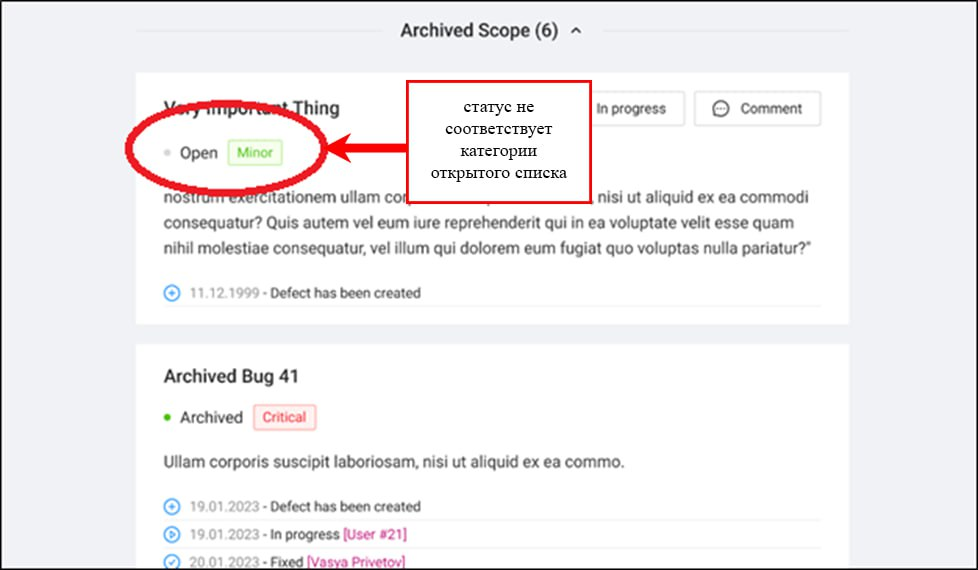


Рисунок 4.2 — Демонстрация дефекта D – 3

По результатам выполнения функционального тестирования был сформирован отчет, о качестве программного продукта на момент тестирования, представленный в таблице 4.3.

Таблица 4.3 – Отчет о выполнении функционального тестирования

|  |  |
| --- | --- |
| Общая оценка качества | Smoke тестирование: пройдено |
| Качество приложения: высокое |
| Рекомендации по модулям | |
| Модуль | Рекомендации |
| Изменение информации о ПП | Обратить внимание на область отображения и редактирования информации о ПП |
| Изменение дефекта | Обратить внимание на процесс изменения дефектов |
| Просмотр архива | Обратить внимание на область архивирования отслеживаемых дефектов ПП |
| Рекомендации | |
| Ни один из найденных дефектов не останавливает работу приложения. Найденные дефекты могут несколько затруднить взаимодействие пользователя с системой. Исправление данных дефектов не является трудоемким, однако позволит повысить качество ПП и увеличить степень удовлетворенности пользователя при взаимодействии с СОДПП. | |

Для тестирования эргономичности и удобства пользования ГПИ было решено использовать эвристики Якоба Нильсена — признанного в мировом сообществе, специалиста в области эксплуатационных характеристик веб–узлов; автор термина «разработка эксплуатационных функций с предварительным анализом». Является доктор наук и автором множества книг, статей и научных работ по теме пользовательского опыта, которые широко цитируются и используются в индустрии. [20]

Результаты проведенного тестирования отражены в таблице 4.4.

Таблица 4.4 – Тестирование с использованием эвристики Якоба Нильсена

|  |  |  |
| --- | --- | --- |
| Эвристики Якоба Нильсена | Аналоги СОДПП | СОДПП |
| 1 | 2 | 3 |
| 1 Видимость статуса системы | Предоставляют сообщения подтверждения и уведомления об ошибках. | Предоставляет подтверждения успешных действий и сообщения об ошибках, предоставляет графическое отображение степени завершенности процессов обработки запросов пользователя. |
| 2 Соответствие системой и реального мира | ГПИ аналогов построен по привычным для пользователя паттернам. | ГПИ СОДПП построен по привычным пользователя паттернам, использует стереотипные символы и знакомые пользователю обозначения. |
| 3 Пользовательский контроль и свобода | Позволяют отказаться от ранее выбранных действий и предоставляют возможность свободного перемещения по страницам приложения. | Позволяет отказаться от ранее выбранных действий и предоставляют возможность свободного перемещения по страницам приложения, предоставляет возможность быстрого доступа некоторых действий. |
| 4 Последовательность и стандарты | ГПИ однотипен и использует одни и те же паттерны для схожих действий. | ГПИ однотипен и использует одни и те же паттерны для схожих действий. |

Продолжение таблицы 4.4

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 5 Предотвращение ошибок | Предотвращают ошибки путем предоставления подсказок пользователю. | Предотвращает ошибки путем ограничений, которые не позволяют пользователю установить неправильное значение, очевидности последовательности действий и использования подтверждений перед деструктивными действиями. |
| 6 Узнавание вместо вспоминания | Перегружены информацией. Требуют запоминания определенных последовательностей действий. | Минимизирует необходимость запоминания путем четко выстроенной последовательности действий и предоставления лишь необходимого функционала на экране. |
| 7 Гибкость и эффективность использования | Требуют долгого поиска необходимого функционала ввиду перегруженности ГПИ информацией. | Позволяет выполнять одни и те же функции разными методами, некоторые из которых являются быстрыми вариантами доступа к функционалу. |
| 8 Эстетичный и минималистичный дизайн | ГПИ перегружен данными. | Предоставляет ясный и минималистичный ГПИ. |
| 9 Понимание проблем и их решение | Могут предоставлять нечеткие сообщения о возникших ошибках. | Предоставляет четкие сообщения о возникших ошибках. |
| 10 Справочные материалы | Предоставляют справочные материалы и подсказки. | Предоставлет справочные материалы и подсказки. |

СОДПП успешно интегрирует основные принципы эвристик Нильсена, обеспечивая удобство и понятность интерфейса.

В процессе функционального тестирования были обнаружены незначительные дефекты, все выявленные дефекты были успешно устранены, функционально программный продукт проходит валидацию.

Результаты анализа удобства взаимодействия с приложением утверждают высокий уровень его эргономичности и удобства использования. Проведенная оценка опыта пользователей подтверждает позитивное восприятие системы, что свидетельствует о тщательном исследовании дизайна и структуры приложения в рамках контекста его функциональной легкости. Продукт успешно совместим с принципами высокой эргономичности, что делает его простым в использовании для конечного пользователя. Результаты тестирования также свидетельствуют о выполнении пользовательскими задачами в приложении, что является важным показателем его эффективности в реальных условиях. Кроме того, отмечается высокая скорость освоения клиентской части системы, что способствует быстрой адаптации пользователей к ее функционалу. Это подтверждают готовность программного продукта к успешному выпуску в открытый доступ. Все функциональные требования выполнены в полном объеме в соответствии с предоставленными ранее задачами на реализацию программного продукта.

5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОТСЛЕЖИВАНИЯ ДЕФЕКТОВ ПРОГРАММНЫХ ПРОДУКТОВ

5.1 Характеристика информационной системы

Разрабатываемый в ходе дипломного проекта ПП направлен на упрощение и систематизацию ключевых этапов процесса отслеживания и учета дефектов ПО, что включает частичную автоматизацию следующих задач: мониторинг (отслеживание) и управление дефектами программных продуктов, анализ их воздействия, управление приоритетами, ведение процессов устранения дефектов. Посредством интеграции системы отслеживания дефектов в окружение разработки, организации способны уменьшить количество ошибок, связанных с управлением дефектами, улучшить эффективность, повысить общий уровень автоматизации трудового процесса, а так же облегчить участникам цикла разработки некоторые из типовых должностных задач.

Преимуществами самой СОДПП являются бесплатность доступа, легкая модифицируемость и масштабируемость, а так же простота интеграции в рабочее окружение. Вышеперечисленные преимущества достигаются за счет дальнейшей публикации проекта под лицензией открытого исходного кода. Написание фундаментальной структуры и функциональности на популярном стеке технологий позволит сторонним разработчикам легко вносить изменения, добавлять новые функции и модули, а также оптимизировать работу приложения под конкретные потребности пользователей или бизнес-процессы организации.

Разрабатываемый ПП предоставит базовое программное решение, изначально способное покрыть запросы малых компаний, а в дальнейшем позволит удовлетворить обще-рыночную потребность путем наращивания необходимого для этого функционала.

СОДПП предоставит возможность структурированного хранения, обработки, изменения и просмотра актуальной информации, необходимой для функционирования отдела тестирования предприятия или другого отдела, схожего по назначению.

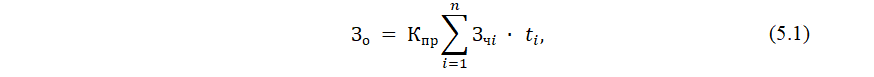
Разрабатываемая система создается для дальнейшего предоставления в бесплатное пользование и будет свободно доступна для всех заинтересованных пользователей. В соответствии с данным назначением производятся все дальнейшие расчеты.

5.2 Расчет сметы затрат на разработку информационной системы

В ходе анализа предметной области было решено, что для реализации СОДПП необходимо привлечь следующих специалистов:

* инженера-программиста (далее — программиста), должностной оклад которого 2000 руб./мес. и трудоемкость выполнения работ 90 ч.;
* тестировщика, должностной оклад которого 1500 руб./мес. и трудоемкость выполнения работ 70 ч.;

Расчет затрат на основную заработную плату команды разработчиков, осуществляется по формуле:



где n – количество исполнителей, занятых разработкой;

– коэффициент премий (составляет 1,5);

– часовой оклад исполнителя i-ой категории, руб.;

– трудоемкость работ, выполняемых исполнителем i-й категории, ч;

Трудоемкость проекта была оценена в 90 часов. Расчет затрат на основную заработную плату команды разработчиков представлен в таблице 5.1.

Таблица 5.2.1 – Расчет затрат на основную заработную плату разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, руб. | Часовой оклад, руб. | Трудоемкость работ, ч. | Итого, руб. |
| Программист | 2 000 | 11,90 | 90 | 1071,00 |
| Тестировщик | 1 500 | 8,90 | 70 | 623,00 |
| Итого | | | | 1694,00 |
| Премия и иные стимульные выплаты (60%) | | | | 1016,40 |
| Итого затраты на основную заработную плату разработчиков | | | | 2710,40 |

Дополнительная заработная плата команды при разработке программного обеспечения включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате:

(5.1)

где – затраты на основную заработную плату;

– норматив дополнительной заработной платы (20%).

Согласно формуле (7.2), затраты на дополнительную заработную плату команды составляют 542,08 руб.

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определены в соответствии с действующими законодательными актами по формуле

(5.3)

где – норматив отчислений в фонд социальной защиты населения и обязательное страхования, где отчисления в фонд социальной защиты составляют 34%, а обязательное страхование – 0,6%.

Согласно формуле (7.3), размер отчислений в фонд социальной защиты населения и на обязательное страхование составляет рублей.

Расчет прочих затрат осуществлен в процентах от затрат на основную заработную плату разработчиков по формуле

(5.4)

где – норматив прочих расходов (30%).

Подставив значения в формулу, сумма прочих затрат составит:

Общая сумма затрат на разработку и реализацию определяется по формуле:

(5.5)

Таблица 5.2.2 включает в себя перечень всех подсчитанных в данном подразделе затрат на разработку СОДПП.

Таблица 5.2.2 – Затраты на разработку программного обеспечения

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Основная заработная плата команды разработчиков | 2710,40 |
| Дополнительная заработная плата команды разработчиков | 542,08 |
| Отчисления в фонд социальной защиты населения и на обязательное страхование | 1125,36 |
| Прочие затраты | 813,12 |
| Общая сумма затрат на разработку | 5190,96 |

Сумма затрат на разработку СОДПП составит 5190,96 руб. Основными статьями расходов являются заработная плата работников и отчисления на отчисления в фонд социальной защиты населения и на обязательное страхование.

5.3 Расчет экономического эффекта от использования информационной системы

Интеграция разрабатываемого продукта, а так же продуктов выстроенных, на его основе приведет к получению неэкономического эффекта. Это выражается в уменьшении количества ошибок, находящихся в разрабатываемом пользователем ПО, повышении точности информации о дефектах этого ПО, а так же ростом общего уровня автоматизации трудового процесса.

Экономия на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ определяется по формуле

(5.6)

где – коэффициент премий;

‒ трудоемкость выполнения работ сотрудниками до и после

внедрения программного средства;

‒ часовая тарифная ставка сотрудника, использующего программное средство;

– плановый объем работ, выполняемых сотрудником;

– норматив дополнительной заработной платы;

– ставка отчислений от заработной платы, включаемых в себестоимость (34,6 %).

Дальнейшие расчеты будут производиться с условием интеграции разрабатываемого ПП в его первичной форме, то есть лишь с определенным в прошлых разделах перечнем функционала. Уровень снижение трудоёмкости выполнения работ основывается на примерах внедрения схожего с СОДПП программного обеспечения в трудовой процесс компаний коммерческой разработки и в среднем составляет около 20 процентов.

По формуле 7.5 рассчитаем экономию на заработной плате и начислениях на заработную плату за счет снижения трудоемкости работ за месяц сотрудника отдела тестирования предприятия:

= 1035,00 руб.

Поскольку система разрабатывается для одного определенного типа сотрудников и в значительной степени затрагивает лишь его, годовая экономия за счет снижения трудоемкости будет рассчитываться следующим образом:

= 12420,00 руб.

Экономическим эффектом при использовании программного средства является прирост чистой прибыли, полученный за счет экономии на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ, который рассчитывается по формуле

где Этек – экономия на текущих затратах при использовании

программного средства;

– прирост текущих затрат, связанных с использованием программного средства;

– ставка налога на прибыль согласно действующему

законодательству;

– затраты на совершенствование программного средства,

является основной заработной платой, руб. (см. таблицу 5.2.2)

Использую значения, полученные ранее, и приравняв прирост текущих затрат к нулю, ввиду бесплатности СОДПП, рассчитаем значение по формуле 5.7:

Оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли по формуле

где – прирост чистой прибыли, полученный от разработки

программного средства организацией-разработчиком по индивидуальному заказу, руб.;

Зр – затраты на разработку программного средства организацией-разработчиком, руб.

Произведем расчет по формуле 5.8:

В результате технико-экономического обоснования были получены следующие значения:

* общая сумма затрат на разработку составила 5190,96 руб.;
* общая годовая экономия за счет снижения трудоемкости работ составила 12420,00 руб.;
* экономическая эффективность разработки более 14 процентов.

Таким образом, разработка информационной системы является эффективной и выделение средств на ее реализацию является целесообразным.

Необходимо понимать, что все из произведенных в данном разделе пояснительной записки расчеты действительны в условиях предоставления в свободный доступ базового решения, и не учитывают ПП, построенные на его основе. Очевидно, что реализация системы с открытым исходным кодом подразумевает тот факт, что любой модуль данной системы может быть модифицирован по усмотрению свободных разработчиков и реализован на рынке как совместно, так и отдельно от базового модуля. Отметим, что последнее является достаточно распространенной практикой в условиях современного рынка.

6 ОХРАНА ТРУДА. ОБОСНОВАНИЕ ВЫБОРА СИСТЕМЫ ОПОВЕЩЕНИЯ О ПОЖАРЕ В ПРОИЗВОДСТВЕННОМ ПОМЕЩЕНИИ

Целью данного дипломного проекта является разработка информационной системы, предназначенной для эффективного отслеживания и управления дефектами в программном коде, а также ее эргономическое обеспечение. Разрабатываемая система представляет собой базовое программное решение, направленное на частичную автоматизацию мониторинга дефектов программных продуктов, упрощение анализа их воздействия, управления приоритетами, ведения их устранения.

Информационная система предназначена для использования работниками отдела тестирования ПО, или других отделов предприятия, схожего по назначению.

В процессе разработки ПП разработчики оказываются в среде, где электроника присутствует в больших объемах, и возможны периодические сбои, вызванные различными внешними факторами, включая технические неисправности, ошибки проводных подключений или воздействие каких либо других внешних условий. Эти неполадки могут привести к возникновению пожаров или других чрезвычайных ситуаций, которые могут иметь серьезные последствия для здоровья работников и безопасности предприятия в целом.

Основными причинами пожаров в таких условиях являются [21]:

* искрение в электрических аппаратах;
* токи коротких замыканий и значительные перегрузки проводов и обмоток электрических устройств, вызывающих их нагрев до высокой температуры;
* плохие контакты в местах соединения проводов, приводящие к увеличению переходного сопротивления;
* халатное и неосторожное обращение с эклектическими приборами.

В свете этого, необходимо уделить особое внимание безопасности рабочего места и принять меры предосторожности, чтобы снизить риск возгорания или иных аварийных ситуаций. В этом контексте, рассмотрение и внедрение систем пожарного оповещения становится неотъемлемой частью оперативного обнаружения и реагирования на потенциальные угрозы подобного типа.

Системы пожарной сигнализации состоят из пожарных извещателей (датчиков), линий связи, приемной станции, откуда сигнал о пожаре может передаваться в помещения пожарных команд [21].

Электрическая пожарная сигнализация в зависимости от схемы соединения извещателей с приемной станцией подразделяется на лучевую и шлейфную. При лучевой схеме от приемной станции к каждому извещателю подводится отдельная проводка, называемая лучом. При шлейфной схеме все извещатели подсоединяются последовательно в один общий провод, оба конца которого подводятся к приемной станции.

В некоторых помещениях может включаться несколько таких проводов или шлейфов, а в один шлейф может быть включено до 50 извещателей.

Пожарные извещатели могут быть ручные (кнопки, установленные в коридорах или лестничных клетках) и автоматические, которые преобразуют неэлектрические физические величины (излучение тепловой и световой энергии, движение частиц дыма) в электрические сигналы определенной формы, передаваемые по проводам на приемную станцию.

По принципу действия извещатели могут быть тепловые (биметаллические, термопарные, полупроводниковые), световые, дымовые, ультразвуковые, комбинированные.

Помещения, предусмотренные для разработки данной информационной системы соответствуют описанию помещений «Административные помещения (офисы), помещения с вычислительной техникой» в рекомендациях по выбору типа пожарных извещателей в зависимости от назначения защищаемого помещения и вида пожарной нагрузки [19]. Для таких помещений рекомендован выбор между дымовым и газовым извещателями.

Сравним дымовые и газовые пожарные извещатели:

1. Дымовые извещатели реагируют на наличие дыма, который обычно возникает при начале пожара. Они могут быть оптическими, ионизационными или комбинированными. Эффективны в обнаружении начальных стадий пожара, когда в помещении еще нет открытого пламени, но уже есть дым. Они подходят для помещений с электронным оборудованием, где возможно возникновение пожара без открытого огня.
2. Газовые извещатели реагируют на наличие определенных газов или паров в воздухе, которые могут свидетельствовать о пожаре или опасной ситуации. Примерами могут быть детекторы угарного газа, детекторы газов для обнаружения утечек газа. Эффективны в обнаружении определенных типов пожаров, таких как пожары, вызванные газовыми утечками или химическими реакциями. Могут обнаруживать опасные газы, которые не всегда сопровождаются дымом или пламенем.

Исходя из этого, можем сделать вывод о необходимости установки дымовых пожарных извещателей. Помимо установки таковых существует необходимость установки ручных пожарных извещателей, позволяющих людям непосредственно обнаруживать пожар и немедленно сообщать о нем службе безопасности или пожарной службе. Это создает дополнительный уровень защиты в случае, если газовые пожарные извещатели не сработают из-за отсутствия дыма или газов, или если возникнет другая неисправность.

Ручные извещатели могут обеспечить быстрое и прямое оповещение о пожаре, минуя возможные задержки или ошибки, которые могут возникнуть при работе автоматических систем детекции пожара. Это может быть особенно важно в ситуациях, когда даже небольшая задержка может привести к серьезным последствиям. Так же, ручные извещатели могут быть использованы не только для обнаружения пожаров, но и для обнаружения других чрезвычайных ситуаций, таких как нападение, медицинские чрезвычайные ситуации.

Место установки ручных пожарных извещателей для типов помещений, не являющихся производственными или помещениями кабельных сооружений, должно соответствовать следующим предписаниям — «извещатели необходимо размещать вдоль эвакуационных путей (в коридорах, холлах, вестибюлях), у выходов из помещений с массовым пребыванием людей, у выходов на лестничные клетки или на лестничной площадке каждого этажа, у общих эвакуационных выходов наружу из здания, вблизи локальных установок пожаротушения с ручным пуском» [22].

Высота установки ручных пожарных извещаетелей определяется исходя из средних антропометрических данных о населении и составляет полтора метра от пола помещения или поверхности земли, в случае если они установлены снаружи помещения. Расстояние между пожарными извещаетелеями ручного типа внутри здания — не более 50 метров, снаружи — 150 метров [23].

При установке приведенных выш типов извещателей система пожарной сигнализации будет обеспечивать не только передачу сигнала о возможном пожаре на станцию МЧС, но и трансляцию предупреждающего сигнала непосредственно в самом помещении. Это позволит людям в чрезвычайной ситуации оперативно реагировать на угрозу и принимать необходимые меры безопасности.

Подача сигнала в помещение через звуковые и визуальные средства, такие как звуковые сирены, световые индикаторы или дисплеи с информацией о пожаре, поможет привлечь внимание присутствующих людей и организовать эвакуацию в безопасное место. Такая дополнительная информация и оповещение внутри помещения увеличивает шансы на эффективную реакцию и минимизацию рисков в чрезвычайной ситуации, способствуя безопасности всех присутствующих в производственном помещении людей.

Исходя из информации приведенной выше в разделе, можно прийти к выводу о необходимости установки двух типов пожарных извещателей: ручных и автоматических газовых. Их необходимо устанавливать в рабочее помещение в соответствии с рекомендациями профессиональных пожарных инженеров, местными строительными нормами, а так же с инструкциями по установке, приведенными на самом изделии.

В составе автоматики на защищаемых объектах следует предусматривать оборудование система передачи извещений, обеспечивающее передачу сигналов о пожаре и неисправности пожарной автоматики на пункт диспетчеризации пожарной автоматики Министерства по чрезвычайным ситуациям Республики Беларусь.

В целом, установка комплексной системы пожарной сигнализации с трансляцией предупреждающего сигнала в помещении играет ключевую роль в обеспечении безопасности как для сотрудников, так и для посетителей здания. Предупреждение о пожаре в реальном времени внутри помещения помогает сократить время реакции и организовать эвакуацию, что способствует минимизации рисков и предотвращению потенциальных угроз для жизни и здоровья людей при разработке ПО данного дипломного проекта.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта разработана информационная СОДПП и решены перечисленные далее задачи.

Проведен анализ предметной области и обзор аналогов. Проведено исследование рынка ПП и выявлены основные недостатки существующих аналогов СОДПП.

Было проведено эргономическое проектирование информационной системы, в ходе которого было разработано:

* алгоритмов работы пользователя;
* эргономические требования к системе;
* создан один из вариантов вида ГПИ системы, удовлетворяющий составленным требованиям;
* сценарий информационного взаимодействия в нескольких формах;
* произведена оценка результатов проектирования.

В ходе разработки системы была выстроена структурная схема СОДПП, сформирована модель БД MongoDB, реализована серверная и клиентская часть системы, сделана оценка произведенных работ. Реализация полностью покрывает и расширяет функционал, поставленных в ходе анализа предметной области, задач. Разработаны инструменты для дальнейшего усовершенствования и программного расширения самой системы.

Проведено тестирование программного средства. В результате тестирования системы была разработана рабочая документация в стандартизированном формате, проведено несколько видов тестирования. Дефекты, выявленные в ходе тестирование ПО были исправлены. Поведение системы соответствует заявленным к ней требованиям.

В результате технико-экономического обоснования были получены следующие значения:

* общая сумма затрат на разработку составила 5190,96 руб.;
* общая годовая экономия за счет снижения трудоемкости работ составила 12420,00 руб.;
* экономическая эффективность разработки более 14 процентов.

Таким образом, было выявлено, что разработка информационной системы является эффективной и выделение средств на ее реализацию является целесообразным.

В разделе по охране труда был обоснован выбор систем оповещения о пожаре.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Андрианов, М. С. Невербальная коммуникация / Андрианов М.С. — Москва: Институт Общегуманитарных Исследований, 2007 — С. 91 – 92
2. Когаловский, М. Р. Энциклопедия технологий баз данных / Когаловский М. Р. — Москва: Финансы и статистика, 2002 — 800 с.
3. Состояние открытого исходного кода. Статистика [Электронный ресурс] — Режим доступа: https://github.blog/2023-11-08-the-state-of-open-source-and-ai
4. Фонд Свободного ПО (FSF) [Электронный ресурс] — Режим доступа: https://www.gnu.org/philosophy/free-sw.html
5. Bugzilla: документация проекта [Электронный ресурс] — Режим доступа: https://www.bugzilla.org/docs/4.2/en/html
6. Atlassian: документация проекта [Электронный ресурс] — Режим доступа: https://www.atlassian.com/legal/software-license-agreement
7. Trac: документация проекта [Электронный ресурс] — Режим доступа: https://trac.edgewall.org/wiki/TracGuide
8. Котлер, Ф. Маркетинг, менеджмент: анализ, планирование, внедрение, контроль / Ф. Котлер; пер. с англ. — СПб.: Питер, 1999 — 896 с.
9. Дормашев, Ю.Б. Общая психология. Введение. В 3 т. / Дормашев Ю.Б, Капустин С.А., Петухов В.В. — Москва: Когито-Центр, 2013 — Т. 1. — 688 с.
10. Шупейко, И.Г. Эргономическое проектирование систем «человек–компьютер – среда» / Шупейко И.Г. — Минск: БГУИР, 2012 — 92 с.
11. ГОСТ Р ИСО 9241-151-2014 Эргономика взаимодействия человек-система. Часть 151: Руководство по проектированию пользовательских интерфейсов сети Интернет / Федеральное агентство по техническому регулированию и метрологии — Москва: Стандартинформ, 2015 — 75 с.
12. ГОСТ ЕН 894-2-2002 Эргономические требования по конструированию средств отображения информации и органов управления. Часть 2: Средства отображения информации / Межгосударственный совет по стандартизации, метрологии и сертификации — Минск: Госстандарт Республики Беларусь, 2004 — 22 с.
13. Раскин, Д. Интерфейс: новые направления в проектировании компьютерных систем / Д. Раскин. — Москва: Символ-плюс, 2005. — 179 с.
14. SPA приложения [Электронный ресурс] — Режим доступа: https://goo.su/pCjh
15. React.js: официальная документация проекта [Электронный ресурс] — Режим доступа: https://react.dev/reference/react/hooks
16. Electron.js: официальная документация проекта [Электронный ресурс] — Режим доступа: https://www.electronjs.org/docs
17. Разница NoSQL и SQL БД [Электронный ресурс] — Режим доступа: <https://www.mongodb.com/nosql-explained/nosql-vs-sql>
18. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. / Шнайер Б. — Москва: Триумф, 2002 — 217 с.
19. Тестирование: тест–кейсы [Электронный ресурс] – Режим доступа: http://www.protesting.ru/testing
20. Консультант по пользовательским интерфейсам Якоб Нильсен [Электронный ресурс] – Режим доступа: https://clck.ru/3ABPNX
21. Михнюк, Т.Ф. Охрана труда. Учебное пособие для вузов. / Михнюк Т.Ф. — Минск: ИВЦ Минфин, 2009 — 358 с.
22. ТКП 45-2.02-317-2018 Пожарная автоматика зданий и сооружений. Строительные нормы проектирования / Министерство архитектуры и строительства Республики Беларусь — Минск: Минстройархитектуры, 2018 — 85 с.
23. СП 5.13130.2009 Системы противопожарной защиты. Установки пожарной сигнализации и пожаротушения автоматические Нормы и правила проектирования / ФГУ ВНИИПО МЧС России — Москва: МЧС России, 2009 — 114 c

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программы

*Фрагмент кода из файла App.jsx:*

*import React, { createContext, useState } from "react";*

*import Header from "./components/header/header";*

*import './styles/style.scss';*

*import { Route, Routes, useNavigate} from "react-router-dom";*

*import SignInSingUp from "./pages/SignInSingUp/SignInSingUp.jsx";*

*import MainView from "./pages/MainView/MainView.jsx";*

*export const GlobalContext = createContext();*

*function App() {*

*const navigate = useNavigate();*

*const [isLoggedIn, setLoggedIn] = useState(false);*

*const serverUrl = 'http://localhost:5555';*

*return (*

*<GlobalContext.Provider value={{serverUrl}}>*

*<div className="App">*

*<Header></Header>*

*<Routes>*

*<Route path="/" element={isLoggedIn ? <MainView /> : <SignInSingUp onLogin={() => setLoggedIn(true)} />}/>*

*<Route path="/login" element={<SignInSingUp onLogin={() => {*

*setLoggedIn(true)*

*navigate('/');*

*}}/>}/>*

*</Routes>*

*</div>*

*</GlobalContext.Provider>);*

*}*

*export default App;*

*Фрагмент кода из файла electron.jsx:*

*const path = require('path');*

*const { app, BrowserWindow, session } = require('electron');*

*const electronIpcMain = require('electron').ipcMain;*

*const isDev = app.isPackaged ? false : require('electron-is-dev');*

*const { importProfiles, exportProfiles, exportProfilesArray } = require('../src/model/profilemgr');*

*const bcrypt = require('bcryptjs');*

*let win, defaultSession;*

*async function createWindow() {*

*win = new BrowserWindow({*

*width: 1920,*

*minWidth: 960,*

*minHeight: 540,*

*height: 1080,*

*resizable: true,*

*titleBarOverlay: true,*

*title: 'trackING sOftware defecTs syStem (СОДПП/INGOTS)',*

*frame: false,*

*webPreferences: {*

*nodeIntegration: true,*

*preload: path.join(\_\_dirname, "preload.js"),*

*},*

*});*

*win.loadURL(*

*isDev*

*? 'http://127.0.0.1:3000'*

*: `file://${\_\_dirname}/../build/index.html`*

*);*

*}*

*app.on('ready', () => {*

*defaultSession = session.defaultSession;*

*createWindow();*

*});*

*app.on('window-all-closed', () => {*

*if (process.platform !== 'darwin') {*

*app.quit();*

*}*

*});*

*app.on('activate', () => {*

*if (BrowserWindow.getAllWindows().length === 0) {*

*createWindow();*

*}*

*});*

*electronIpcMain.on('trafficlight-channel', (e, data) => {*

*if (data.action === 'minimize') win.minimize();*

*if (data.action === 'close') win.close();*

*if (data.action === 'maximize') {*

*win.isMaximized() ? win.unmaximize() : win.maximize();*

*}*

*});*

*electronIpcMain.on('export-profiles-channel', async (e, data) => {*

*if (data.action === 'export-profiles') {*

*exportProfiles(data.profilesData || null);*

*}*

*});*

*electronIpcMain.handle('duplex-profiles-channel', async (e, data) => {*

*try {*

*const profiles = await importProfiles();*

*switch (data.action) {*

*case 'search':*

*const profile = profiles.find(obj => (obj.credentials.login === data.credentials.login));*

*if (!profile) return new Error('Such profile doesnt exist', { cause: { login: false, password: false }});*

*if (profile.credentials.password) {*

*//console.log(data.credentials.password, profile.credentials.password);*

*return await bcrypt.compare(data.credentials.password, profile.credentials.password) || new Error('Invalid password', { cause: { login: true, password: false }});*

*}*

*return null;*

*case 'export-profiles-append-array':*

*return new Promise(async (resolve, reject) => {*

*try {*

*let newProfiles = [];*

*if (typeof data.profilesData === 'object' && !Array.isArray(data.profilesData)) {*

*newProfiles = [data.profilesData];*

*} if (Array.isArray(data.profilesData)) {*

*newProfiles.push(...data.profilesData);*

*} else {*

*reject('Invalid profilesData format. Expecting an object or an array of objects.');*

*return;*

*}*

*const password = newProfiles[0].credentials.password;*

*newProfiles[0].credentials.password = await bcrypt.hash(password, (await bcrypt.genSalt(10)));*

*exportProfilesArray(newProfiles || null)*

*.then((result) => {*

*resolve(null);*

*})*

*.catch((error) => {*

*console.error('Error during exportProfilesArray:\n', error);*

*reject(error);*

*});*

*} catch (error) {*

*console.error('Error in export-profiles-channel:\n', error);*

*reject(error);*

*}*

*});*

*default: return null;*

*}*

*} catch (error) {*

*console.log(error.message);*

*return error;*

*}*

*});*

*electronIpcMain.handle('set-cookies-channel', (e, cookiesArray) => {*

*if (!Array.isArray(cookiesArray)) cookiesArray = [cookiesArray];*

*if (!cookiesArray.every(cookie => typeof cookie === 'object')) {*

*return new Error('Invalid data format. Expecting an array of objects.');*

*}*

*for (const cookieData of cookiesArray) {*

*const { key, value, expires } = cookieData;*

*const options = {*

*httpOnly: true,*

*secure: false,*

*path: '/',*

*};*

*defaultSession.cookie.set({ name: key, value, expires, ...options })*

*}*

*return { success: true };*

*});*

*Фрагмент кода из файла DefectManagmentView.jsx:*

*import React from 'react';*

*import styles from './DefectManagmentView.module.scss';*

*import { useContext, useEffect, useState } from 'react';*

*import { Badge, Form, Select, Table } from 'antd';*

*import { GlobalContext } from '../../App';*

*import PageContainer from '../../components/PageContainer/PageContainer';*

*import PageHeader from '../../components/PageHeader/PageHeader';*

*import Pagination from '../../components/Pagination/Pagination';*

*import Search from 'antd/es/input/Search';*

*import Link from 'antd/es/typography/Link';*

*function getBadgeStatus(type\_of\_state) {*

*switch (type\_of\_state) {*

*case "open":*

*return "default";*

*case "in\_progress":*

*return "processing";*

*case "fixed":*

*return "success";*

*default:*

*return "default";*

*}*

*}*

*const columns = [*

*{*

*title: 'Status',*

*dataIndex: 'current\_state',*

*key: 'status',*

*render: (state) => state ?*

*<Badge status={getBadgeStatus(state.type\_of\_state)}*

*className={styles.badge}*

*text={(state.type\_of\_state.toUpperCase()[0] +  state.type\_of\_state.slice(1)).replace('\_', ' ')}/>*

*: null,*

*width: 120,*

*},*

*{*

*title: 'Defect',*

*dataIndex: 'defect\_title',*

*key: 'defect',*

*render: (text, record) => <Link ellipsis={{rows: 1}}>{text}</Link>,*

*width: '25%',*

*},*

*{*

*title: 'Project',*

*dataIndex: 'project',*

*key: 'project',*

*render: (project) => project ? <Link ellipsis={{rows: 1}}>{project.project\_title}</Link> : null,*

*responsive: ['xl']*

*},*

*{*

*title: 'Priority',*

*dataIndex: 'priority',*

*key: 'priority',*

*render: (priority) => priority ?*

*<Badge status='error' text={priority.toUpperCase()[0] +  priority.slice(1)}/>*

*: null,*

*width: 200,*

*responsive: ["sm"]*

*},*

*{*

*title: 'Severity',*

*dataIndex: 'severity',*

*key: 'severity',*

*render: (e) => e ?*

*<Badge status='error' text={e.toUpperCase()[0] +  e.slice(1)}/>*

*: null,*

*width: 200,*

*responsive: ["sm"]*

*},*

*{*

*title: 'Action',*

*dataIndex: 'action',*

*key: 'action',*

*width: 140,*

*}*

*];*

*const DefectManagmentView = () => {*

*const [defects, setDefects] = useState([]);*

*const [loading, setLoading] = useState(false);*

*const [totalAmount, setTotalAmount] = useState(1);*

*const [searchQuery, setSearchQuery] = useState("");*

*const { serverUrl } = useContext(GlobalContext);*

*const sizeOfPage = 11;*

*useEffect(() => {*

*fetchData(1, sizeOfPage);*

*}, []);*

*const fetchData = (page, pageSize, query = '') => {*

*setLoading(true);*

*const url = `${serverUrl}/defects?page=${page}&limit=${pageSize}&search=${query}`;*

*fetch(url, {*

*method: 'GET',*

*headers: {*

*'Content-Type': 'application/json',*

*"Accept": "application/json",*

*}*

*})*

*.then(response => {*

*if (!response.ok) {*

*throw new Error('Network response was not ok');*

*}*

*return response.json();*

*})*

*.then(data => {*

*setDefects(data.defects);*

*setTotalAmount(data.count);*

*setLoading(false);*

*})*

*.catch(error => {*

*console.log(error);*

*});*

*};*

*return (<div className={styles.DefectManagmentView}>*

*<PageHeader subtitle={'Track defects and projects'}>Dashboard</PageHeader>*

*<PageContainer className={styles.content}>*

*<div className={styles['filter-container']}>*

*<div className={styles['search-subcontainer']}>*

*<Search placeholder="Search"*

*onSearch={(value) => {*

*setSearchQuery(value);*

*fetchData(1, sizeOfPage, value)*

*}}/>*

*</div>*

*<Form className={styles['filter-subcontainer']} layout="inline">*

*<Form.Item label="Priority">*

*<Select style={{width: 140}} placeholder="Any priority">*

*<Select.Option value="High"></Select.Option>*

*<Select.Option value="Medium"></Select.Option>*

*<Select.Option value="Low"></Select.Option>*

*</Select>*

*</Form.Item>*

*<Form.Item label="Severity">*

*<Select style={{width: 140}} placeholder="Any severity">*

*<Select.Option value="Critical"></Select.Option>*

*<Select.Option value="Major"></Select.Option>*

*<Select.Option value="Average"></Select.Option>*

*<Select.Option value="Minor"></Select.Option>*

*</Select>*

*</Form.Item>*

*<div className={styles['button-container']}>*

*<button>Clear</button>*

*<button>Apply Filters</button>*

*</div>*

*</Form>*

*</div>*

*<Table*

*rowKey={e => e.\_id}*

*scroll={{ y: 627 }}*

*className={styles.table}*

*loading={loading}*

*columns={columns}*

*dataSource={defects}*

*expandable={{*

*expandedRowRender: (record) => (*

*<p*

*style={{*

*margin: 0,*

*fontSize: 14,*

*marginLeft: 50,*

*}}*

*>*

*{record.description}*

*</p>*

*),*

*rowExpandable: (record) => record.name !== '',*

*}}*

*pagination={false}>*

*</Table>*

*<Pagination*

*className={styles['pagination-bar']}*

*totalCount={totalAmount}*

*pageSize={sizeOfPage}*

*fetchData={(page) => fetchData(page, sizeOfPage, searchQuery)}*

*serverUrl={serverUrl}*

*/>*

*</PageContainer>*

*</div>);*

*};*

*export default DefectManagmentView;*

*Фрагмент кода из файла electron.jsx:*

*import styles from './MainView.module.scss';*

*import { useState } from 'react';*

*const items = [*

*{*

*label: 'Defects Management',*

*key: 'defects',*

*},*

*{*

*label: 'Project Management',*

*key: 'projects',*

*},*

*{*

*label: 'About',*

*key: 'about',*

*},*

*];*

*const MainView = () => {*

*const [current, setCurrent] = useState('defects');*

*const onClick = (e) => setCurrent(e.key);*

*return (*

*<div className={styles['main-wrapper']}>*

*<HorizontalMenu onClick={onClick} selectedKeys={[current]} items={items}></HorizontalMenu>*

*<DefectManagmentView></DefectManagmentView>*

*</div>*

*);*

*}*

*export default MainView;*

*Фрагмент кода из файла Pagination.jsx:*

*import React, { useState } from 'react';*

*import classnames from 'classnames';*

*import { usePagination, DOTS } from './usePagination';*

*import styles from './Pagination.module.scss';*

*const Pagination = props => {*

*const {*

*totalCount,*

*siblingCount = 1,*

*pageSize,*

*className,*

*fetchData,*

*} = props;*

*const [currentPage, setCurrentPage] = useState(1);*

*const paginationRange = usePagination({*

*currentPage,*

*totalCount,*

*siblingCount,*

*pageSize*

*});*

*if (currentPage === 0 || paginationRange.length < 2) {*

*return null;*

*}*

*const onNext = () => {*

*setCurrentPage(currentPage + 1);*

*fetchData(currentPage + 1, pageSize);*

*};*

*const onPrevious = () => {*

*setCurrentPage(currentPage - 1);*

*fetchData(currentPage - 1, pageSize);*

*};*

*let lastPage = paginationRange[paginationRange.length - 1];*

*return (*

*<ul*

*className={classnames(styles['pagination-container'], { [className]: className })}*

*>*

*<li*

*className={classnames(styles['pagination-item'], {*

*[styles.disabled]: currentPage === 1*

*})}*

*onClick={onPrevious}*

*>*

*<div className={classnames(styles.arrow, styles.left)} />*

*</li>*

*{paginationRange.map(pageNumber => {*

*if (pageNumber === DOTS) {*

*return <li className={classnames(styles["pagination-item"], styles['dots'])}>&#8230;</li>;*

*}*

*return (*

*<li*

*className={classnames(styles['pagination-item'], {*

*[styles.selected]: pageNumber === currentPage*

*})}*

*onClick={() => {*

*setCurrentPage(pageNumber);*

*fetchData(pageNumber, pageSize);*

*}}*

*>*

*{pageNumber}*

*</li>*

*);*

*})}*

*<li*

*className={classnames(styles['pagination-item'], {*

*[styles.disabled]: currentPage === lastPage*

*})}*

*onClick={onNext}*

*>*

*<div className={classnames(styles.arrow, styles.right)}/>*

*</li>*

*</ul>*

*);};*

*export default Pagination;*