**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Инженерная школа информационных технологий и робототехники Отделение информационных технологий**

**Информатика и вычислительная техника**

Отчет по Лабораторной работе 6

Issue tracker на примере BitBucket

Выполнил

Студент группы 8В23 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Балашов А.Д.

Проверил

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Хамухин А. А.

Томск 2024

**Цель работы:**

Получить опыт практической работы в паре при создании, разработки ПО, а так же его тестировании и исправлении дефектов.

**Задание**

1. Необходимо протестировать Windows-приложение, написанное в

лабораторной работе №5, для этого студентам необходимо разбиться на пары. Один будет выполнять роль разработчика, другой — специалиста по тестированию.

1. В репозитории необходимо выбрать вашу команду, созданную в

лаб. раб. № 2

1. Необходимо нажать кнопку «Manage team» в левом верхнем углу окна, затем добавить новую группу «Quality engineers» нажав кнопку «Add Group» и выставить уровень доступа «read» этой группы
2. Затем необходимо ввести имя (или e-mail) пользователя, который будет исполнять роль специалиста по тестированию, в блок группы «Quality engineers» и нажать кнопку «Add».
3. Специалист по тестированию выполняет «Clone» проекта, выполненного в рамках лаб. раб. №5. Инструкцию по выполнению клонирования см. в лаб. раб. №2. Затем специалист по тестированию проверяет программу на наличие ошибок, в качестве которых может служить заполнение поля ввода отрицательными или некорректными значениями. Далее специалист по тестированию создает issue типа bug в Issue Tracker (пункт «Issues» в меню на главной странице репозитория), с описание найденных ошибок. В поле Title вводится название ошибки, в поле Description — информация о найденной ошибке, т.е. здесь описываются шаги воспроизведения ошибки, результат, который ожидается в итоге, текущий результат. Пример: вы вводите в поле «Цена» отрицательное значение, но в итоге он получаете сообщение об ошибке, т. к. поле ввода содержит отрицательное значение. В поле «Assignee» необходимо выбрать имя разработка из списка. Также можно задать приоритет в поле Priority, в зависимости от серьезности ошибки. Разработчик должен руководствоваться приоритетом при определении порядка исправления ошибок: сначала должны исправляться ошибки с наибольшим приоритетом. Также необходимо приложить скриншот ошибки, который поможет разработчику локализировать и исправить данную ошибку.
4. Разработчик просматривает найденный дефект в Issue Tracker, можно воспользоваться пунктом меню My Issues (рис. 6.6). Затем разработкой проверяет описанный сценарий и исправляет ошибку в программе, выполняет Commit и Push изменений с помощью TortoiseGit (см. лабораторную работу №2). Затем разработчик заходит в Issue Tracker, наживает кнопку «Resolve» (либо через меню «Workflow») (рис. 6.7–6.8), в поле комментарий вводит номер ревизии (номер внесённых изменений), см. пункт меню «Commits». Также можно выполнить редактирование issue с помощью кнопки «Edit» (рис. 6.8).
5. Специалист по тестированию обновляет свой локальный репозиторий (Pull) и запускает проект. Проверяет дефект, если он исправлен, то ставит статус Closed, в противном случае выполнят Open дефекта с соответствующим комментарием.
6. Специалисту по тестированию необходимо создать как минимум два дефекта в Issue Tracker, затем разработчик и специалист по тестированию меняются ролями.

**Процесс выполнения**

После просмотра приложения и его кода был выявлен ряд проблем, например нет обработчика событий неправильного ввода пользователем. (рисунок 1 и 2)

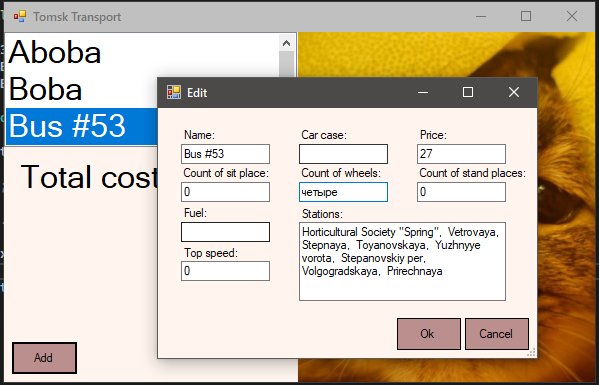


Рисунок 1 – Ввод данных провоцирующий ошибку

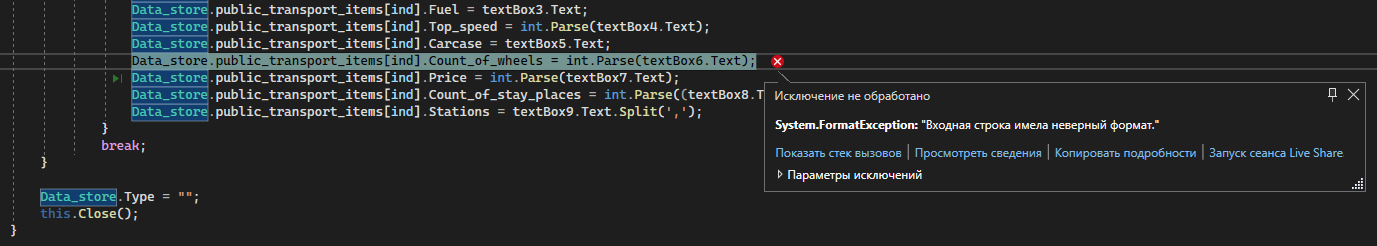


Рисунок 2 – Пример найденной ошибки

Эта проблема наблюдается со всеми формами в этом приложении. Так же нет обработчика на ввод отрицательных чисел. В связи с этим отправляю разработчику доклад об ошибке. (рисунок 3)

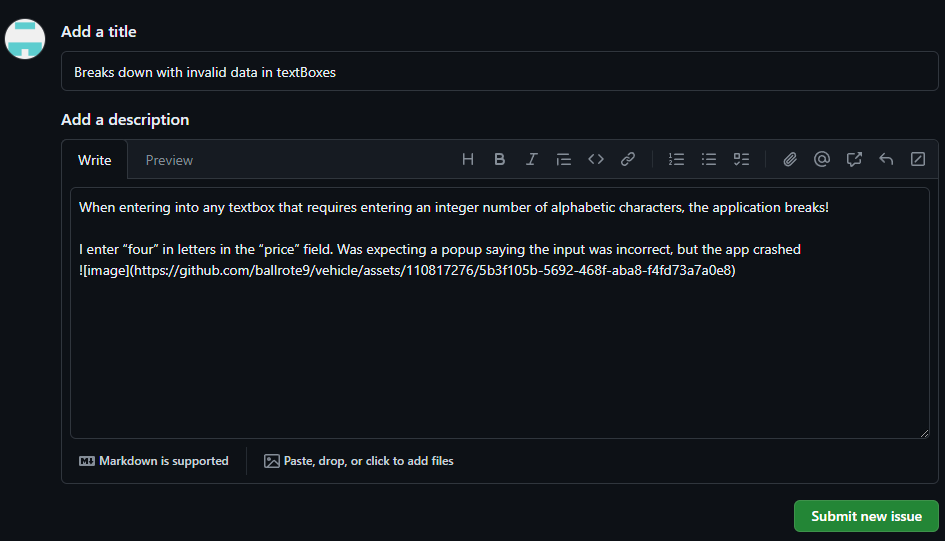


Рисунок 3 – Создание issue

Получаю доклад о найденной ошибке (рисунок 4), внимательно изучаю содержимое сообщения.

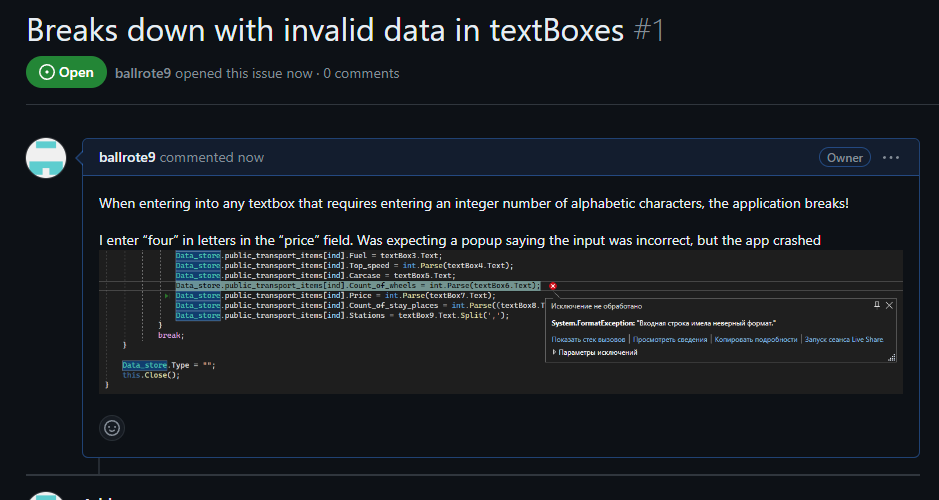


Рисунок 4 – Получение issue

Повторив сценарий, получил ту же самую ошибку. Исправил ее добавлением функции, проверяющей корректность введенных пользователем данных и возвращающей булевое значение. (рисунок 5)

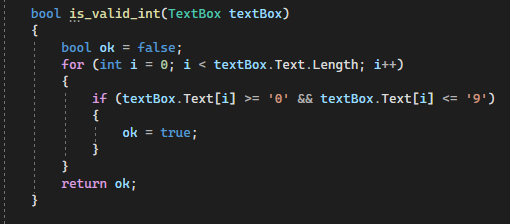


Рисунок 5 – Исправление ошибки

Затем информирую всех, что ошибка была исправлена. (рисунок 6)

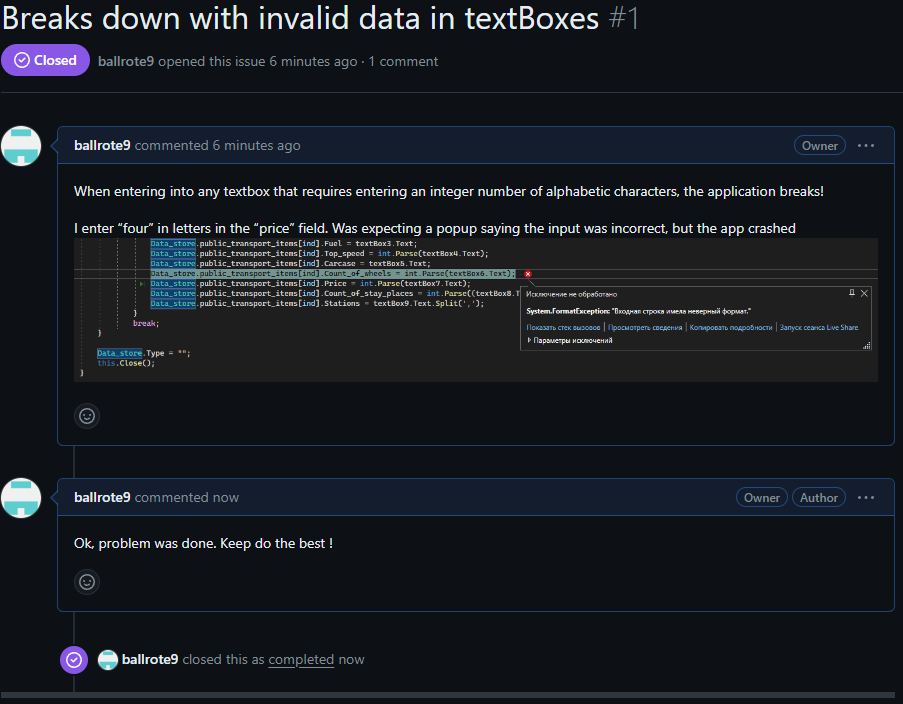


Рисунок 6 – Закрытие issue

Теперь проблема решена, однако была найдена еще одна. А именно открытие спец окон происходит далеко от основной формы, а также их размер может быть катастрофически большим или маленьким. (рисунок 7)

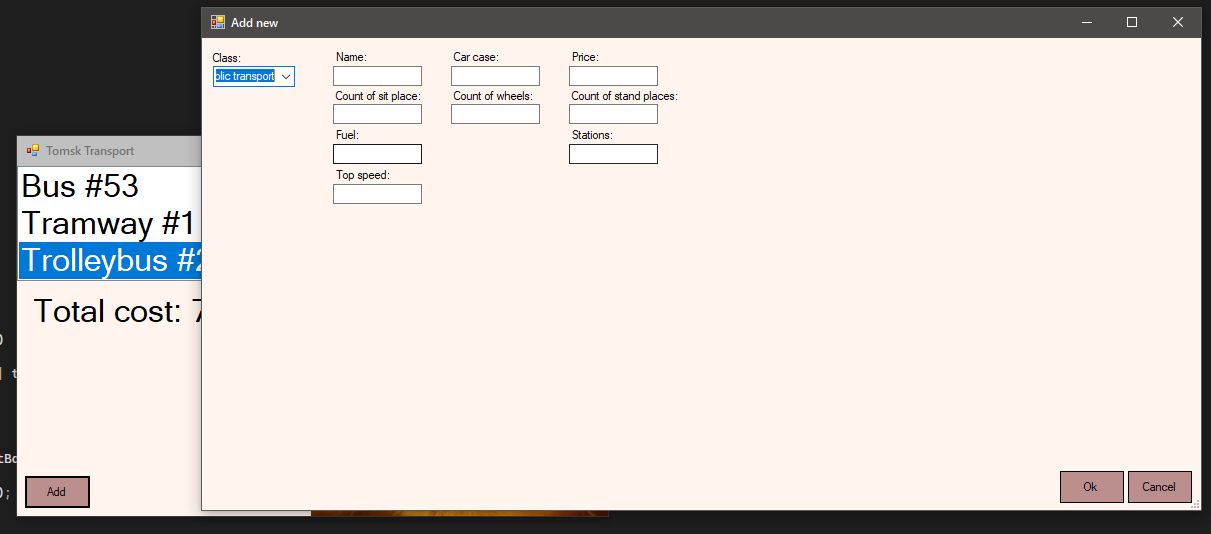


Рисунок 7 – Проблема

Повторяется процедура с созданием issue.

Исправляю ошибку.

**Ответы на контрольные вопросы**

**1. Что такое Issue Tracker? Где и для чего применяется?**

**Issue Tracker** (трекер задач, трекер проблем) — это программный инструмент, который используется для отслеживания и управления проблемами, задачами, ошибками и запросами на улучшение в проектах. Он применяется в различных сферах, таких как разработка программного обеспечения, управление проектами, поддержка клиентов и других областях, где важно организовать работу с задачами и проблемами.

**Применение:**

* **Разработка ПО:** Для отслеживания багов, новых функций, технических долгов и улучшений.
* **Управление проектами:** Для мониторинга выполнения задач и управления проектными рисками.
* **Поддержка клиентов:** Для регистрации и разрешения обращений и запросов от клиентов.
* **Контроль качества:** Для систематического управления дефектами и обеспечения качества продукции.

**2. Что является разновидностью Issue Tracker? (дать определение)**

Разновидностями Issue Tracker являются **Bug Tracker** и **Task Management System**.

* **Bug Tracker (Трекер ошибок):** Это разновидность Issue Tracker, специализированная на отслеживании ошибок (багов) в программном обеспечении. Основная цель — идентифицировать, классифицировать, приоритизировать и отслеживать ошибки до их исправления. Примеры: Bugzilla, MantisBT.
* **Task Management System (Система управления задачами):** Это разновидность Issue Tracker, ориентированная на управление задачами и рабочими процессами. Такие системы помогают организовать и контролировать выполнение задач в рамках проектов, обеспечивая своевременное выполнение и координацию работы команды. Примеры: Asana, Trello, Jira.

**3. Какие сведения может включать в себя база данных об обнаруженных ошибках?**

База данных об обнаруженных ошибках может включать следующие сведения:

* **Идентификатор ошибки:** Уникальный номер или код ошибки.
* **Описание ошибки:** Детальное описание проблемы.
* **Шаги для воспроизведения:** Инструкции для воспроизведения ошибки.
* **Ожидаемый результат:** Что должно было произойти при правильной работе.
* **Фактический результат:** Что произошло на самом деле.
* **Приоритет:** Уровень важности исправления ошибки (например, низкий, средний, высокий, критический).
* **Статус:** Текущий статус ошибки (например, новая, в работе, исправлена, закрыта).
* **Ответственный:** Член команды, назначенный для исправления ошибки.
* **Дата обнаружения:** Когда ошибка была впервые зафиксирована.
* **Дата последнего обновления:** Когда ошибка в последний раз обновлялась.
* **Среда:** Информация о среде, в которой возникла ошибка (операционная система, браузер, версия ПО и т.д.).
* **Комментарии:** Дополнительные замечания и обсуждения.
* **Прикрепленные файлы:** Скриншоты, логи и другие документы, связанные с ошибкой.

**4. Привести пример жизненного цикла дефекта (с пояснениями)**

**Пример жизненного цикла дефекта:**

1. **Создание (New):** Дефект обнаружен и зарегистрирован в системе. Включает описание, шаги для воспроизведения и другую необходимую информацию.
2. **Назначение (Assigned):** Дефект назначен конкретному разработчику или члену команды для дальнейшего анализа и исправления.
3. **Анализ (In Progress):** Разработчик анализирует причину дефекта и разрабатывает план по его исправлению.
4. **Исправление (Fixed):** Дефект исправлен разработчиком. Код изменен и протестирован локально.
5. **Тестирование (Testing):** Исправленный дефект передается тестировщику для проверки. Тестировщик проводит тестирование, чтобы убедиться, что ошибка исправлена и не вызывает новых проблем.
6. **Проверка (Verified):** Тестировщик подтверждает, что дефект исправлен корректно и работает в соответствии с требованиями.
7. **Закрытие (Closed):** Дефект закрыт, так как он исправлен и проверен.
8. **Повторное открытие (Reopened):** Если после закрытия дефекта выясняется, что ошибка не была полностью исправлена или появилась снова, дефект может быть переоткрыт и процесс начинается заново с шага "Назначение".

Эти стадии могут варьироваться в зависимости от используемой методологии и системы отслеживания дефектов.

**Вывод**

В ходе выполнения лабораторной работы был получен опыт практической и теоретической работы в разработке ПО, а так же в его тестировании и исправлении дефектов.

**Список используемых источников:**

1. Scott Chacon. Pro Git [Электронный ресурс]. – Режим доступа: https://gitscm.com/book/ru/v2, свободный.
2. Mike McQuaid. Git in Practice, Manning, 20 14
3. Ron Patton. Software Testing, 2nd ed., Sams Publishing, 2006.
4. Савин Р. Тестирование dot com — М. : Дело, 2007 — 312 с.
5. Котляров В.П. Основы тестирования ПО — М. : Интернет-Ун-т Информ. Технологий, 2006. — 288 с.