Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Курсовая работа.

Дисциплина: «Технологии программирования»

Выполнил: Хлебодаров А.С.

Группа: 6306-030301D

Проверил: Белоусов А.А.

Самара 2022

СОДЕРЖАНИЕ

1. Введение………………………………………………………………….....3
2. План проекта………………………………………………………………..4
3. Сценарии использования…………………………………………………..5
4. Описание архитектуры…………………………………………………….8
5. План тестирования………………………………………………………....9
6. Инструкции пользователя системы……………………………………...10
7. Результаты тестирования…………………………………………………16
8. Код реализации системы…………………………………………………21
9. Заключение………………………………………………………………..22

ВВЕДЕНИЕ

*Trackiti* — инструмент для начинающих разработчиков, позволяющий отслеживать, какие на данный момент имеются баги и фичреквесты от пользователей. Подойдёт преимущественно небольшим командам, которые недавно выкатили свой продукт и начинают развивать и корректировать его.

Проблема состоит в том, что невозможно до релиза проверить работу разработки во всех возможных сценариях, поэтому приложение на ранних этапах требует исправления багов, замеченных пользователями. Также для привлечения новых юзеров и расширения функционала программы нужно добавлять новые фичи и возможности. Так вот, для оперативного выполнения упомянутого необходимо наглядное представление информации, а также доступное в удобном виде для всех членов команды. Поэтому моя программа будет состоять в виде нескольких таблиц, содержащихся в разных вкладках, например: «Баги», «Фичи», «Реализованное». Руководитель команды или ответственный человек сможет размещать и редактировать в них информацию, а остальные члены команды смогут её просматривать для получения сведений о, например, сути задачи, статусе, исполнителе, технических деталях и т.д. Это поможет оптимизировать процесс обслуживания и поддержки новых разработок.

ПЛАН ПРОЕКТА

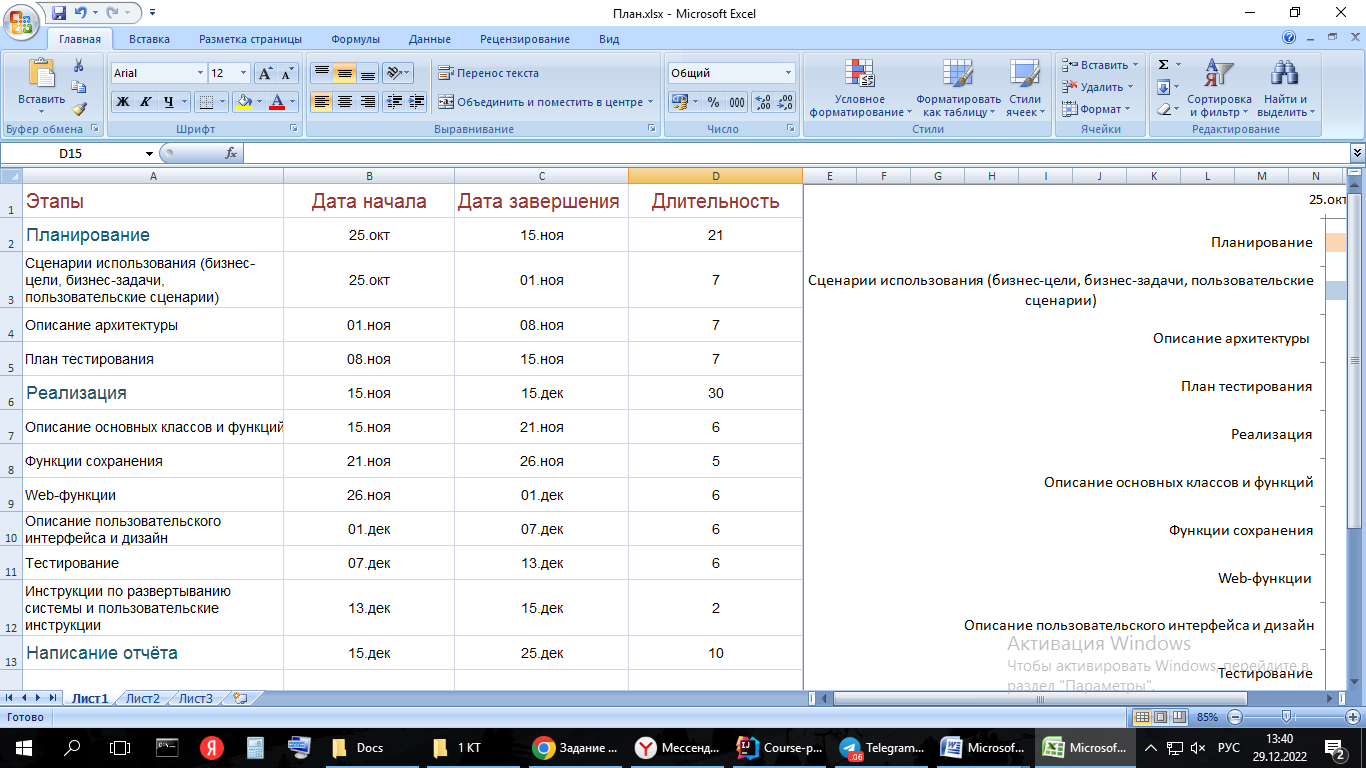


Рисунок 2.1 — Список основных этапов с датами их исполнения

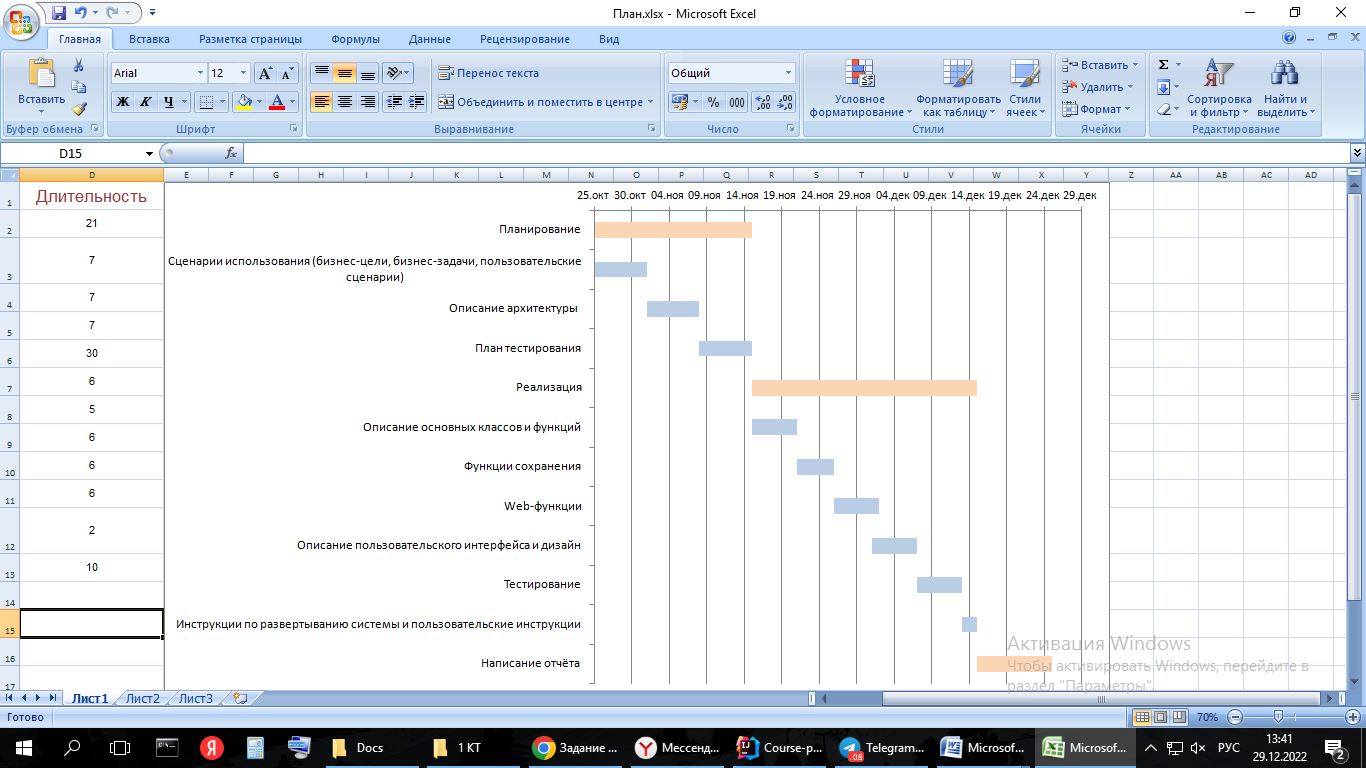


Рисунок 2.2 — План проекта в виде диаграммы Ганта

СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

*Trackiti* — инструмент для начинающих разработчиков, позволяющий отслеживать, какие на данный момент имеются баги и фичреквесты от пользователей.

**Роли пользователей**

* Руководитель
* Участник разработки

**Бизнес-цели**

**BG1 Привлекать новых пользователей в свежее приложение путём исправления багов и добавления в него новых функций**

**F1-1 Организация работы над багами и фичреквестами**

Для того чтобы привлекать новых пользователей в свежее приложение путём исправления багов и добавления в него новых функций

Я как руководитель

Хочу вносить и редактировать в системе информацию о багах и фичреквестах

**UC1-1-1 Добавление нового бага**

Допустим, я зашёл в систему как руководитель

И открыта вкладка «Баги»

Когда нажимаю «Добавить баг»

И заполняю необходимые поля

И нажимаю «Добавить»

Тогда в системе появляется баг со статусом «В работе»

**UC1-1-2 Редактирование бага**

Допустим, я зашёл в систему как руководитель

И открыта вкладка «Баги»

Когда нажимаю «Редактировать баг»

И изменяю необходимые поля

И нажимаю «Сохранить»

Тогда в системе обновляется баг

**UC1-1-3 Добавление нового фичреквеста**

Допустим, я зашёл в систему как руководитель

И открыта вкладка «Фичреквесты»

Когда нажимаю «Добавить фичу»

И заполняю необходимые поля

И нажимаю «Добавить»

Тогда в системе появляется фича со статусом «В работе»

**UC1-1-4 Редактирование фичреквеста**

Допустим, я зашёл в систему как руководитель

И открыта вкладка «Фичреквесты»

Когда нажимаю «Редактировать фичу»

И изменяю необходимые поля

И нажимаю «Сохранить»

Тогда в системе обновляется фича

**F1-2 Получение данных о задаче**

Для того чтобы повышать эффективность своей работы

Я как участник команды

Хочу получать понятную информацию о текущих задачах

**UC1-2-1 Просмотр данных о задаче**

Допустим, я зашёл в систему как участник разработки в начале или в течение рабочего дня

И открыты вкладки «Баги» или «Фичреквесты»

Просматривая поля «Исполнитель», «Суть», «Технические детали», «Дедлайн», «Приоритет» и т.д.

Получаю все необходимые данные для организации предстоящей работы и её успешного выполнения

**BG2 Повышение прибыли путём добавления в разработку только окупающихся функций**

**F2-1 Анализ окупаемости добавленных функций**

Для того чтобы повышать прибыль путём добавления в разработку только окупающихся функций

Я как руководитель

Хочу просматривать в системе информацию о уже реализованном функционале

**UC2-1-1 Добавление реализованного**

Допустим, я зашёл в систему как руководитель

И открыта вкладка «Фичреквесты»

Выбираю нужное, нажимаю «Добавить в Реализованное»

И оно появляется во вкладке «Реализованное»

**UC2-1-2 Просмотр информации о добавленных функциях**

Допустим, я зашёл в систему как руководитель

И открыта вкладка «Реализованное»

Просматривая информацию о реализованных фичах и зная статистику новых пользователей, могу анализировать окупаемость каждой

ОПИСАНИЕ АРХИТЕКТУРЫ

Проект будет представлять собой web-приложение, реализованное на языке Java. Выбрана web-структура, а не мобильное приложение, потому что таким образом программа будет наиболее полезна в предполагаемых области и сценариях использования, где необходимо клиент-серверное взаимодействие пользователей. Язык выбран по следующим причинам:

* Для реализации предполагаемых в программе вкладок, таблиц, кнопок и т.д. лучше всего подходит объектно-ориентированный подход
* Java хорошо подходит для web-приложений
* Этот язык обеспечит большую эффективность работы, так как команда владеет им лучше, чем другими языками программирования.

Проект реализован в IntelliJ IDEA Ultimate Edition. Данная версия этой IDE позволяет использовать такие web-технологии, как JSP, JSTL, сервлеты. В дополнение был использован Apache Tomcat, который обеспечил проект веб-сервером и программной средой для функционирования сервлетов. Для упрощения использования совокупности этих технологий была применена система сборки Maven. Для придания приложению яркого и приятного дизайна была использована библиотека W3.CSS, которая предоставляет широкий набор стилей для документов HTML и JSP.

Структура программы построена на основе паттерна проектирования MVC (model-view-controller, модель-представление-контроллер). В программе присутствуют такие сущности, как Bug, Feature, Realized, реализованные в виде Java-классов. Списки этих сущностей хранятся в соответствующих им моделях, которые реализованы в виде Java-классов на основе паттерна проектирования Singleton. Контроллерами являются соответствующие сервлеты, которые также выступают в качестве посредников между моделями и их представлением в виде JSP страниц.

ПЛАН ТЕСТИРОВАНИЯ

Тестирование будет проводиться путём последовательного запуска частей программы и анализа результатов. Переход к реализации следующей части программы будет происходить после получения удовлетворительных результатов тестирования работы предыдущих, а также их совместимости. После получения в конечном итоге полной сборки программы также будет проводиться её тестирование. Устранение ошибок будет происходить путём их локализации, последующего переписывания кода и нового тестирования.

В частности, необходимо проверить корректность отображения данных о багах, фичреквестах и реализованном в виде таблиц. Также необходимо проверить работу функций по добавлению и редактированию этих элементов. Будет производиться запуск страниц с этим функционалом и тестирование ввода различных данных.

ИНСТРУКЦИИ ПОЛЬЗОВАТЕЛЯ СИСТЕМЫ

При запуске программы открывается стартовая страница (рис 6.1), на которой имеются три кнопки. Они ведут на страницы, относящиеся к багам, фичреквестам и реализованному.

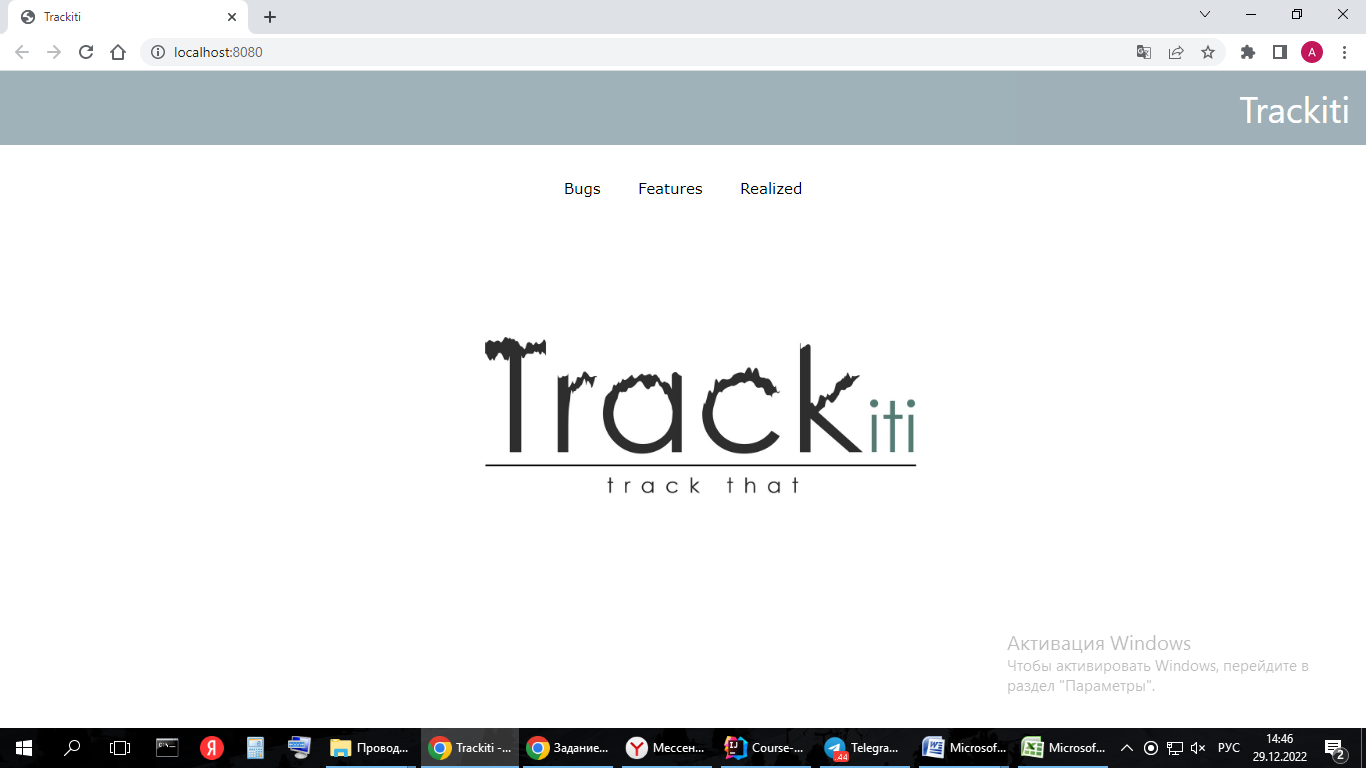


Рисунок 6.1 — Стартовая страница

Если нажать на кнопку «Bugs», то откроется страница с багами (рис.6.2), на которой видны шапка таблицы и кнопки.

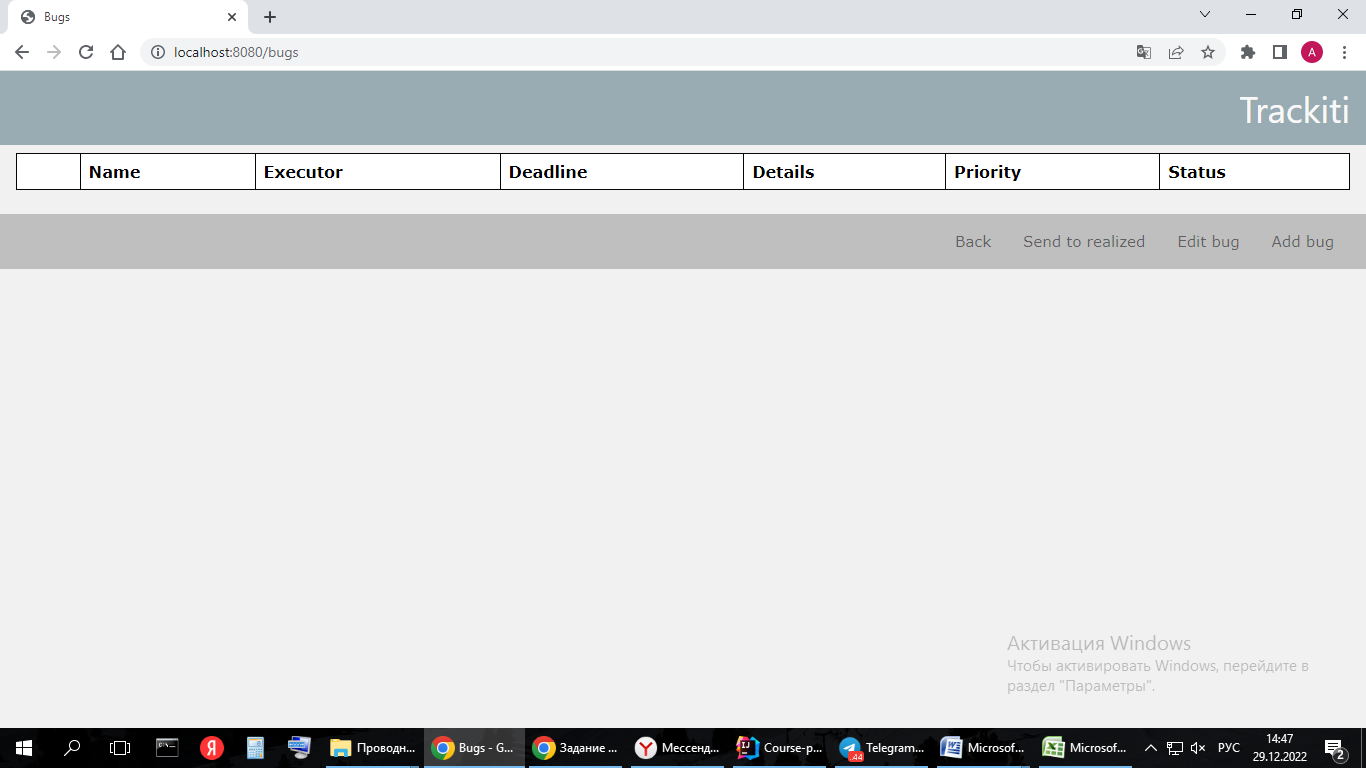


Рисунок 6.2 — Страница багов

Нажав на кнопку «Add bug», можно увидеть поля для заполнения информации о названии, исполнителе, дедлайне, деталях, приоритете и статусе (рис 6.3). Подтверждение введённых данных происходит по кнопке «Submit». Также на большинстве страниц есть кнопка «Back», которая ведёт на предыдущую страницу.

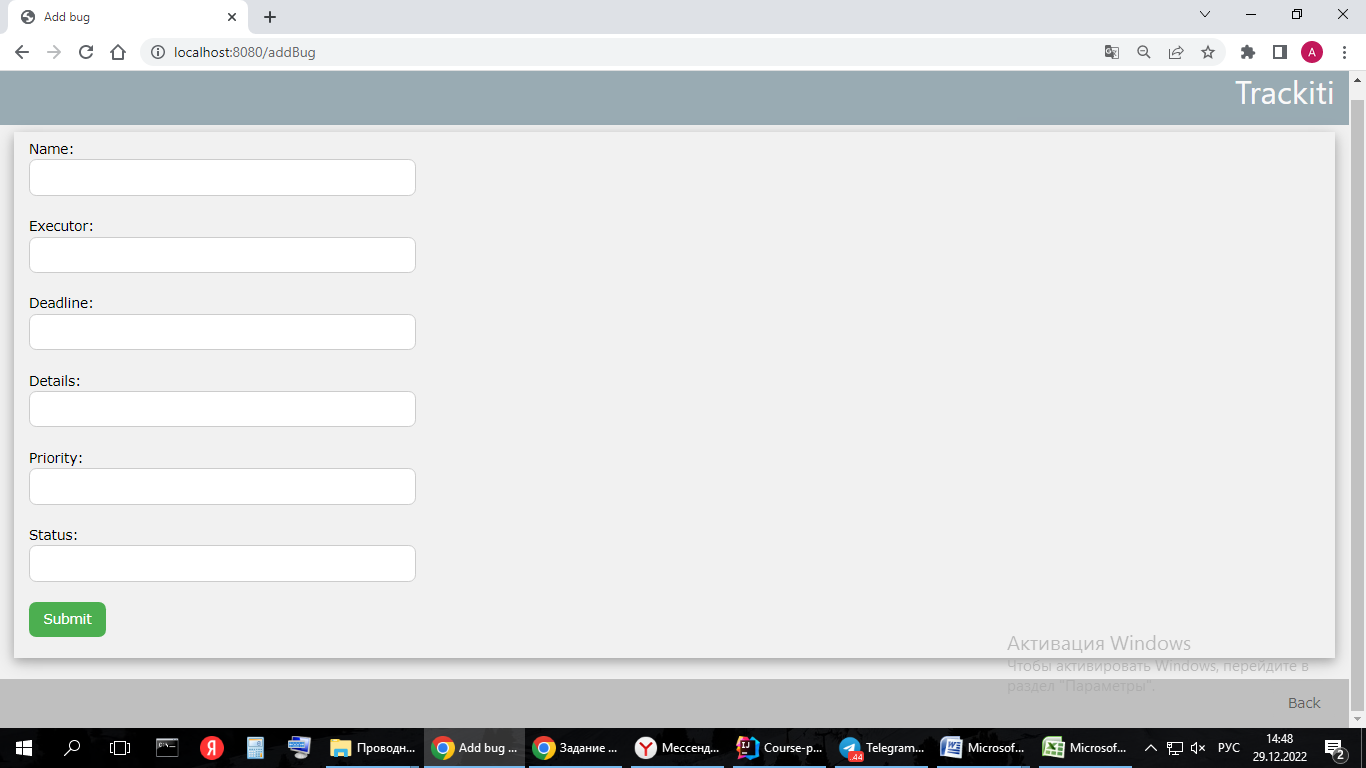


Рисунок 6.3 — Страница с полями ввода данных о баге

Также существует возможность редактирования информации по кнопке «Edit bug» (рис 6.4). На открывшейся странице нужно ввести номер редактируемого бага и заполнить поля, которые необходимо отредактировать.

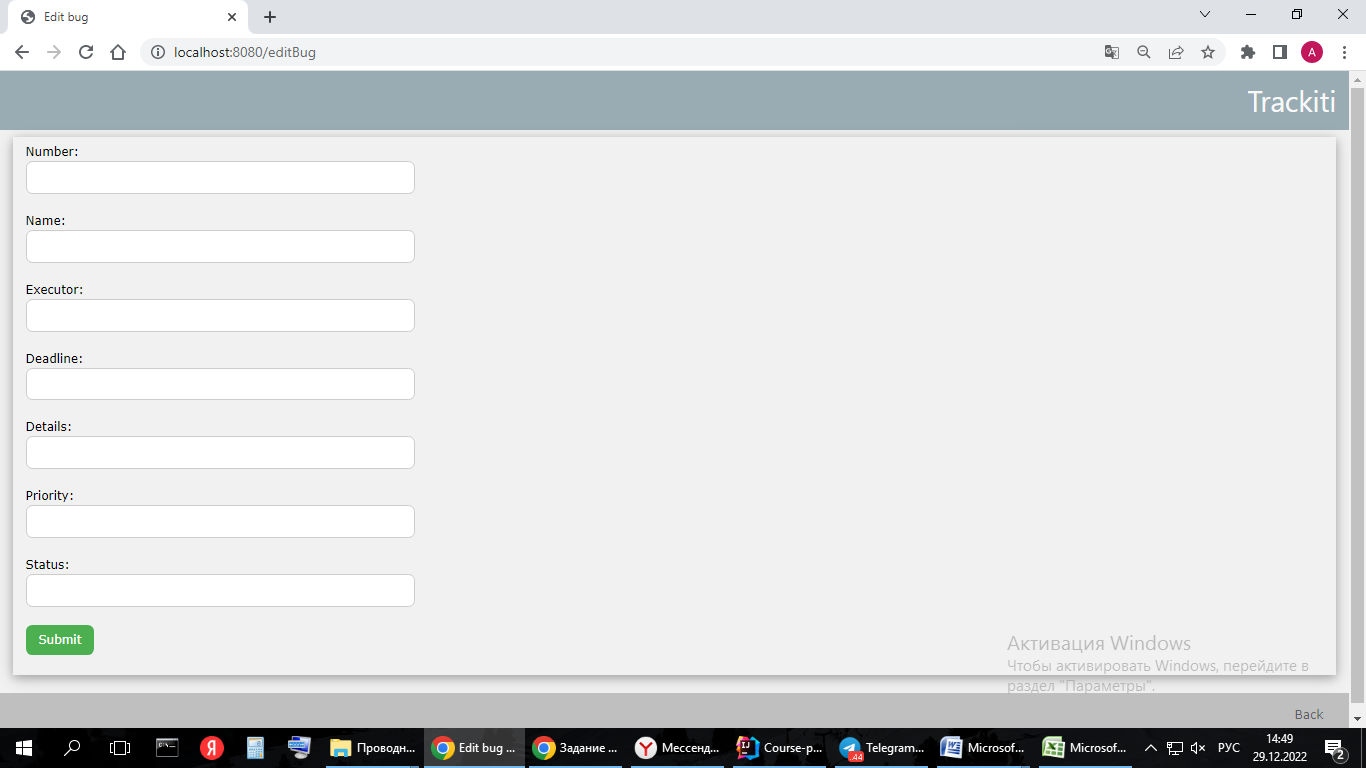


Рисунок 6.4 — Страница редактирования бага

По кнопке «Send to Realized» предоставляется возможность занести баг по номеру в таблицу реализованного (рис 6.5).

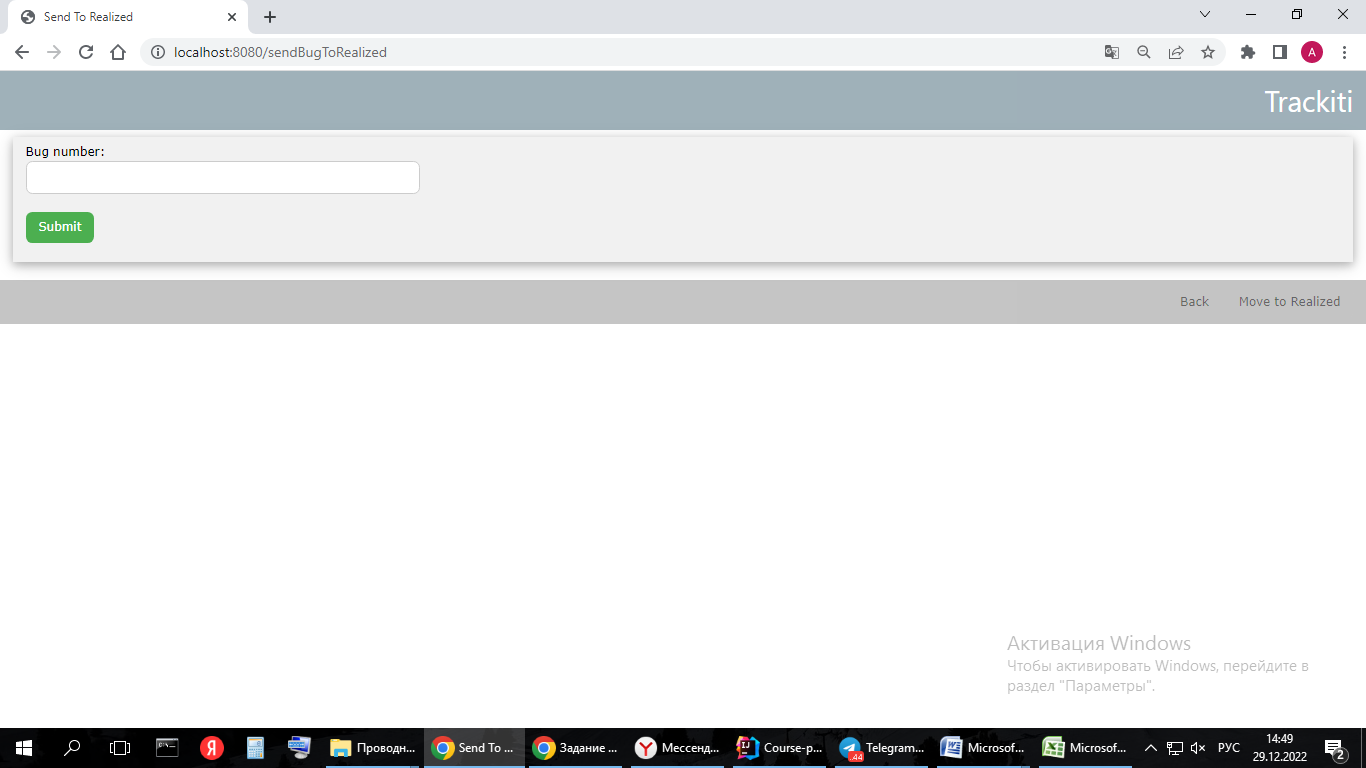


Рисунок 6.5 – Страница для занесения бага в реализованное

На странице фичреквестов представлен аналогичный функционал (рис.6.6 - 6.9).

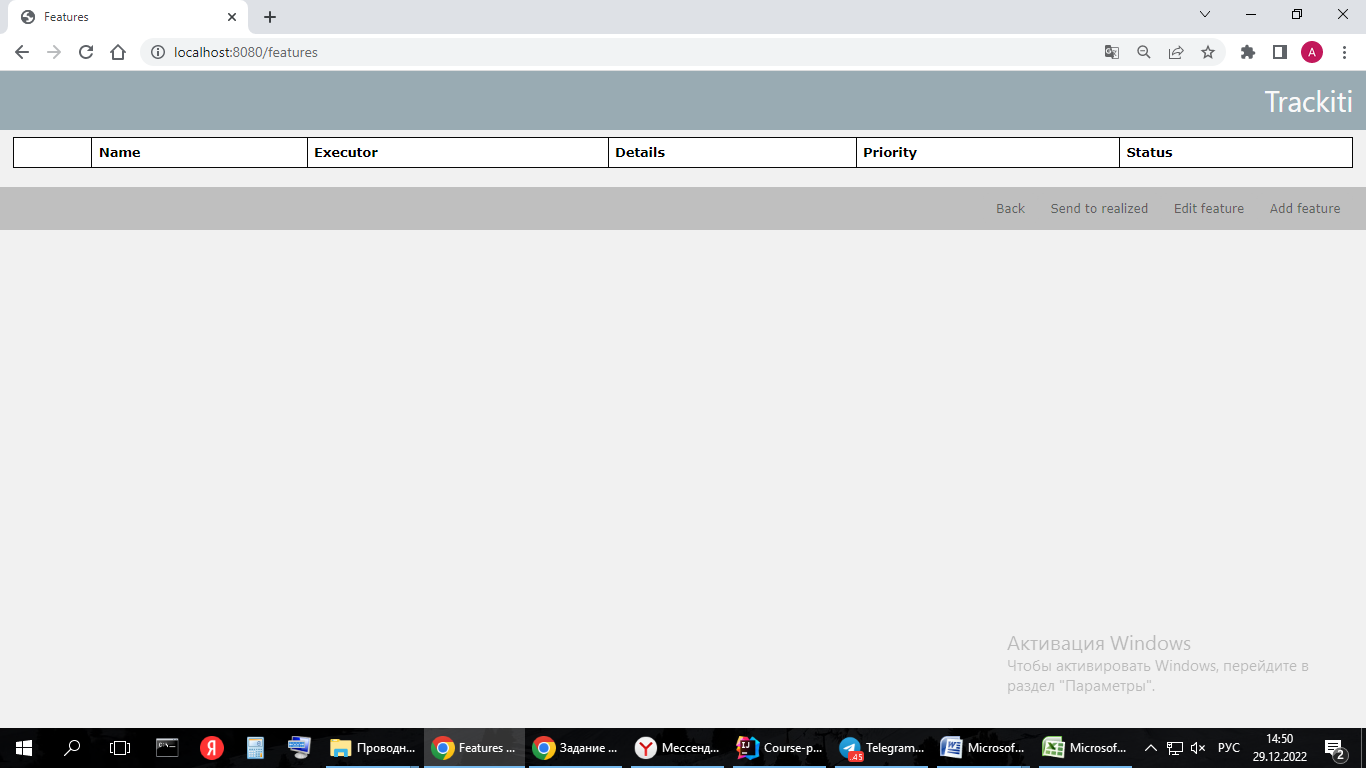


Рисунок 6.6 — Страница фичреквестов

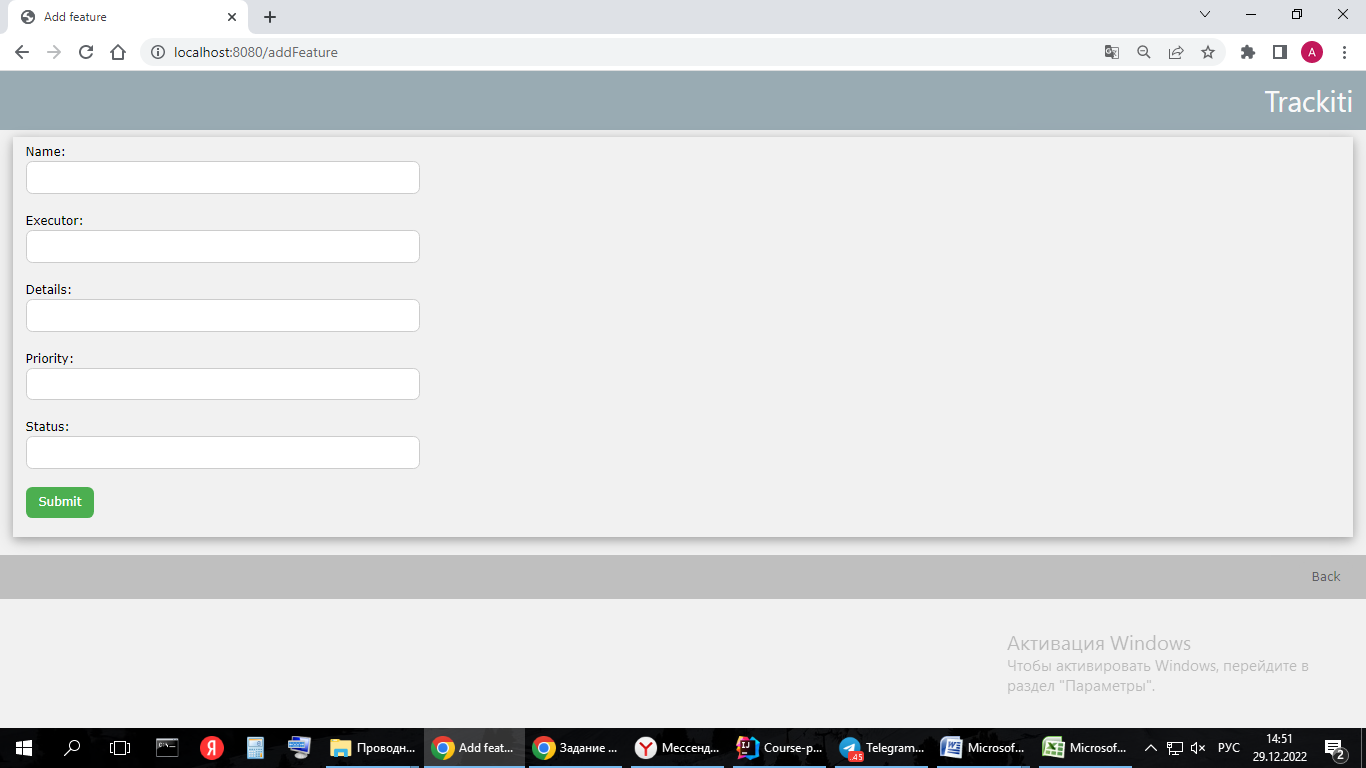


Рисунок 6.7 — Страница с полями ввода данных о фичреквесте

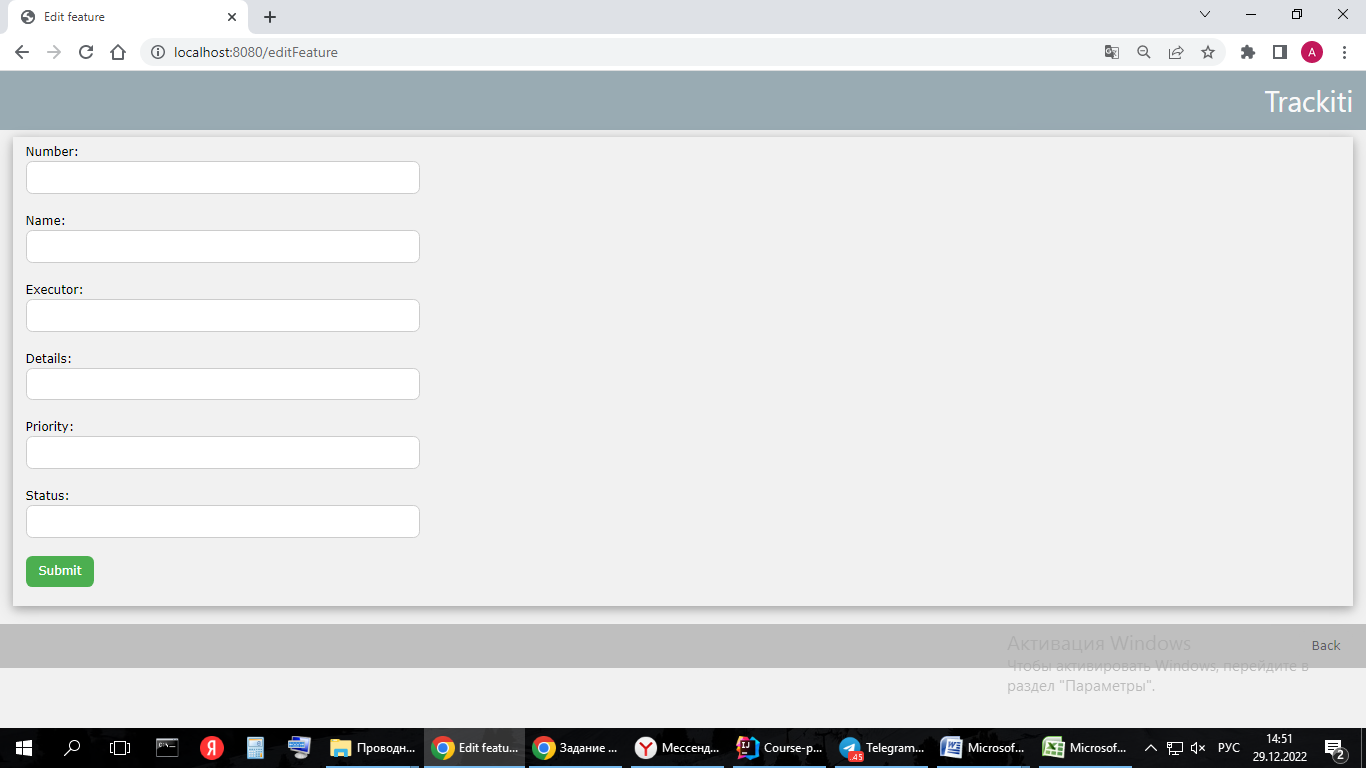


Рисунок 6.8 — Страница редактирования фичреквестов

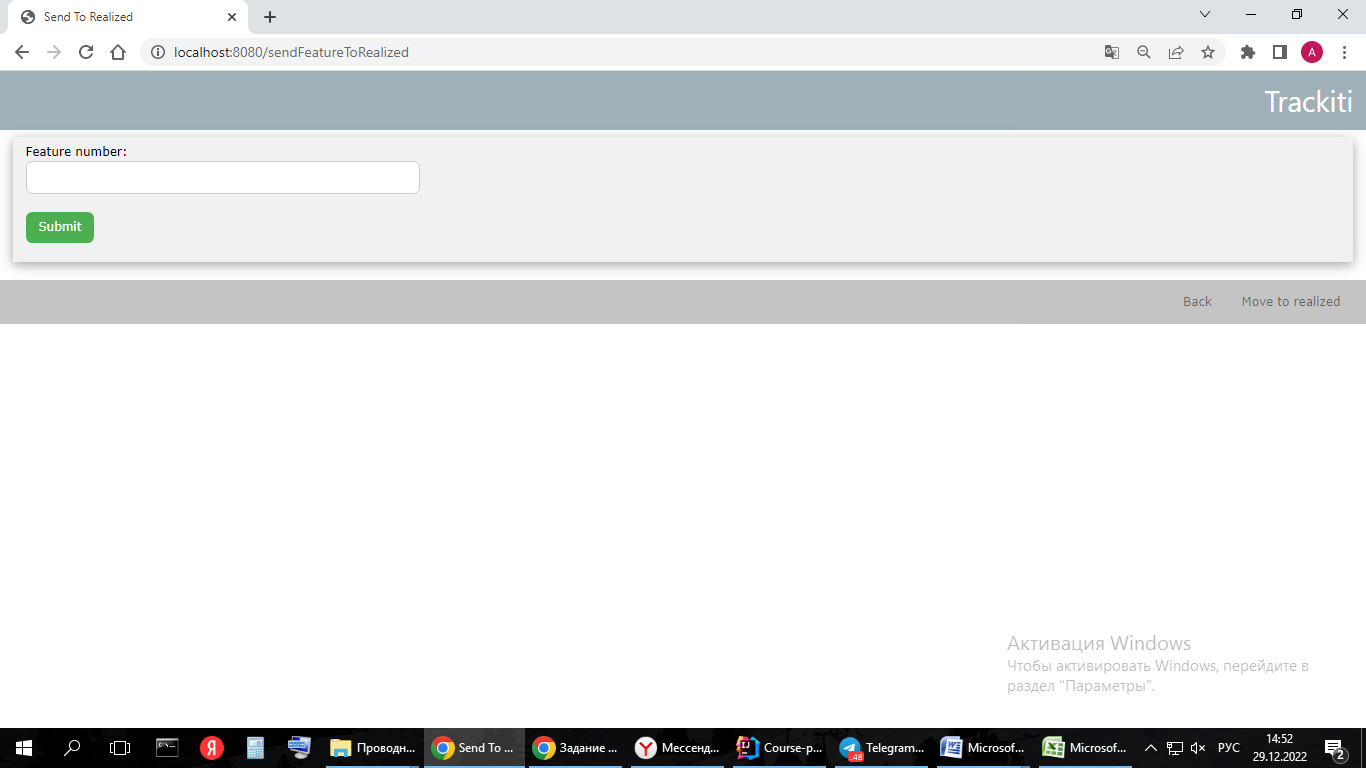


Рисунок 6.9 – Страница для занесения фичреквеста в реализованное

При нажатии кнопки «Realized», находящейся на главной странице, открывается страница с таблицей реализованных багов и фичреквестов (рис.6.10).

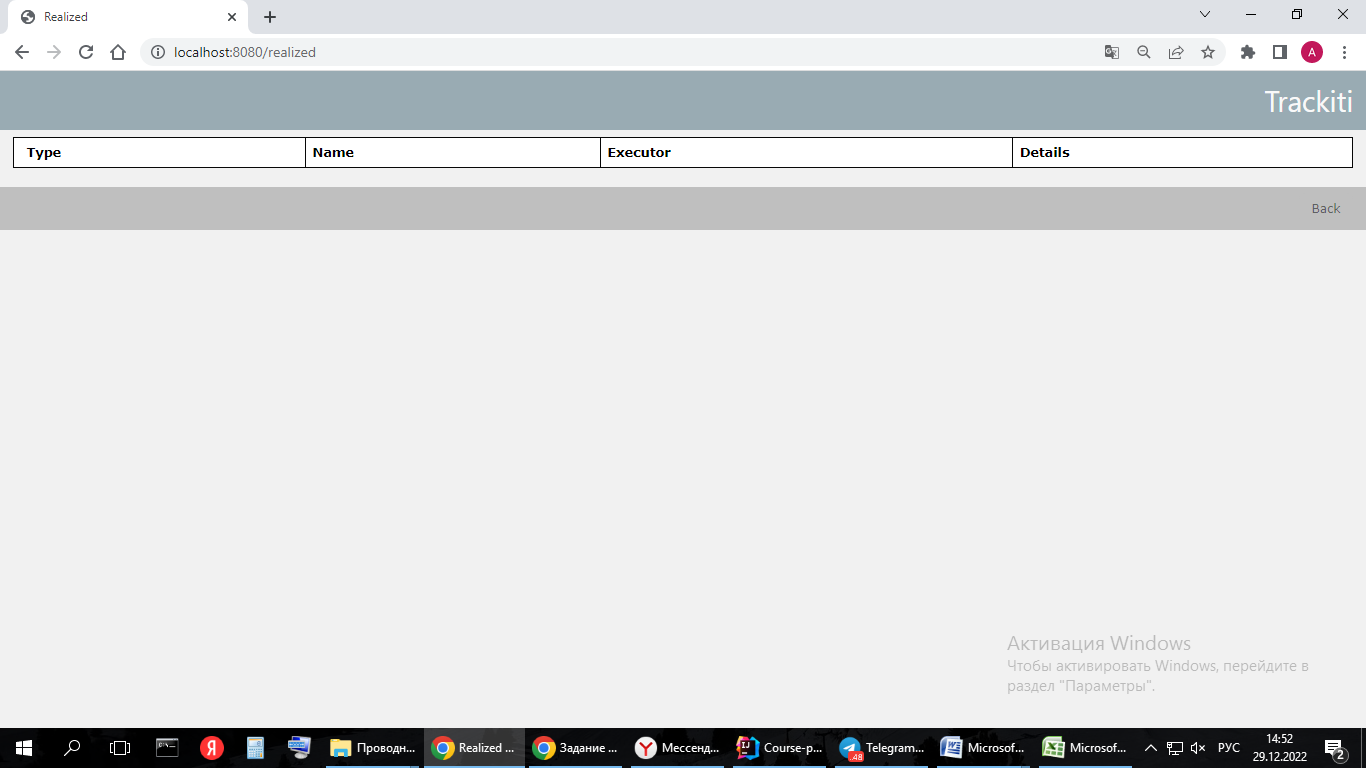


Рисунок 6.10 — Страница реализованного

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Из скриншотов предыдущего раздела видно, что все имеющиеся страницы открываются корректно. Проверим теперь корректность ввода данных и отображения их в таблицах. Добавим пару багов (рис.7.1 – 7.2) и посмотрим, как они будут отображаться (рис.7.3).

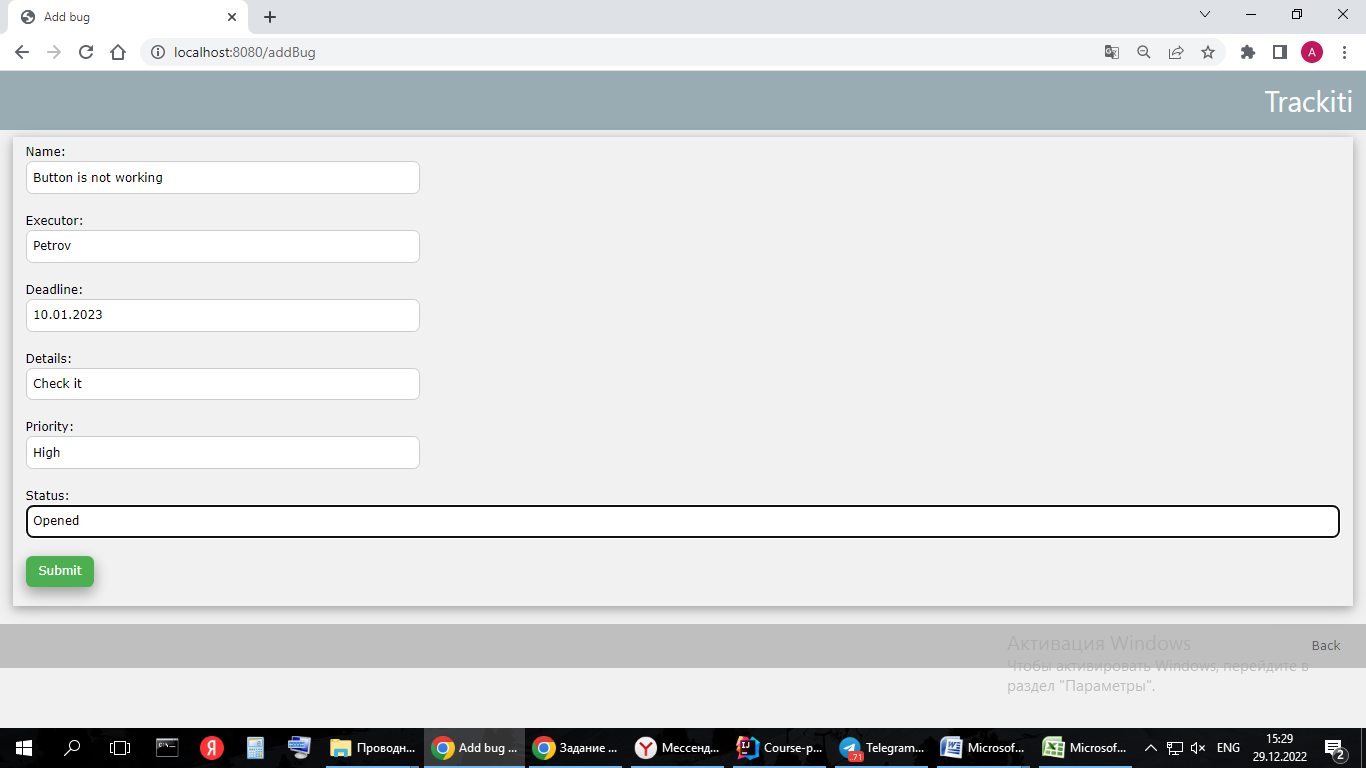


Рисунок 7.1 — Добавление бага

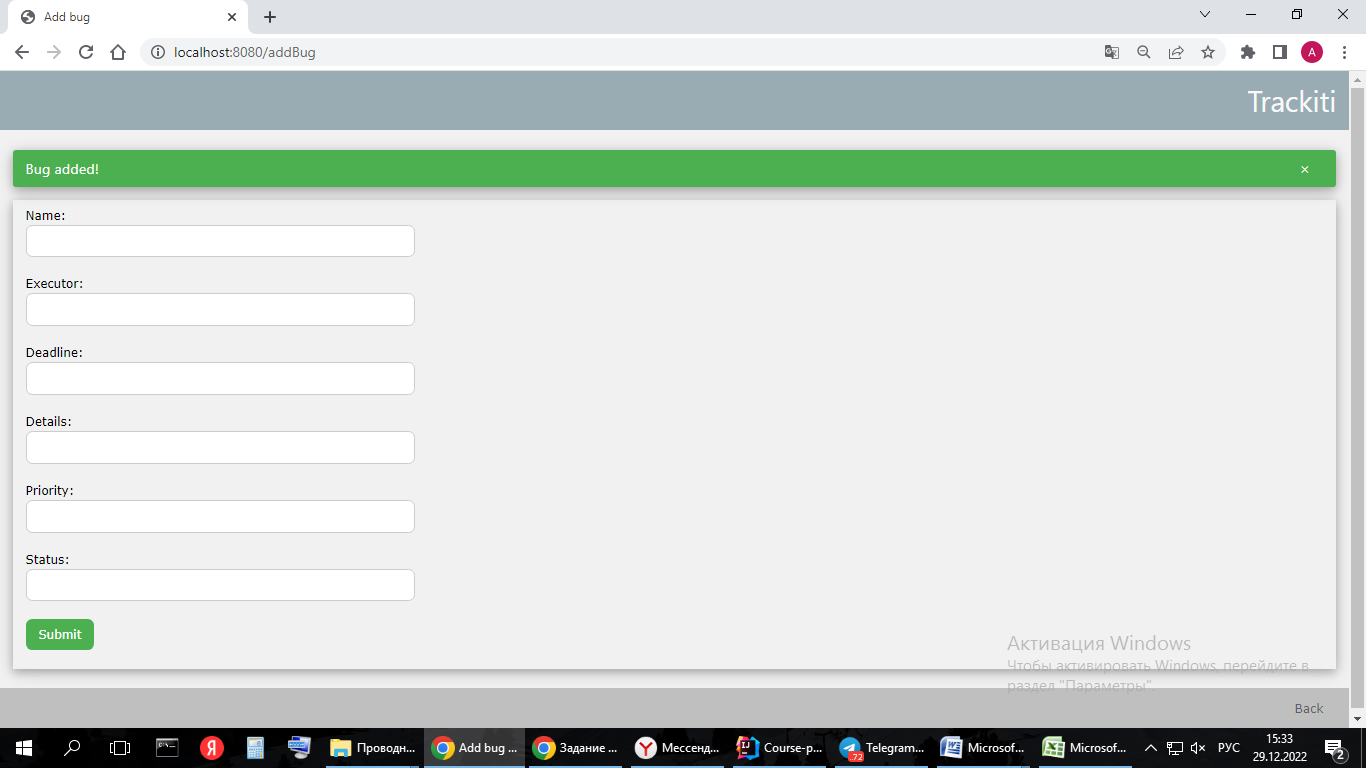


Рисунок 7.2 — Добавление бага

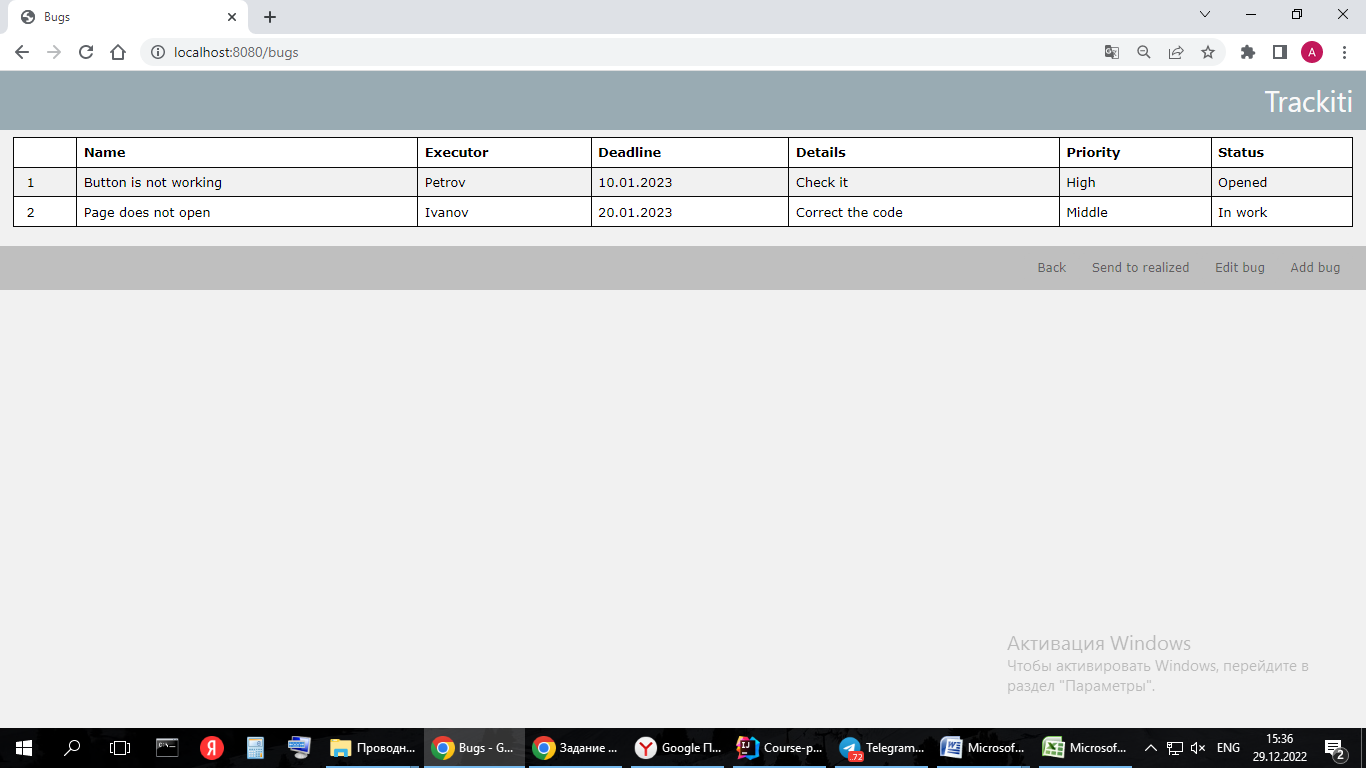


Рисунок 7.3 — Отображение результатов

Как видим — всё отображается корректно. Попробуем отредактировать баг (рис.7.4) и посмотрим на изменения в таблице (рис.7.5).

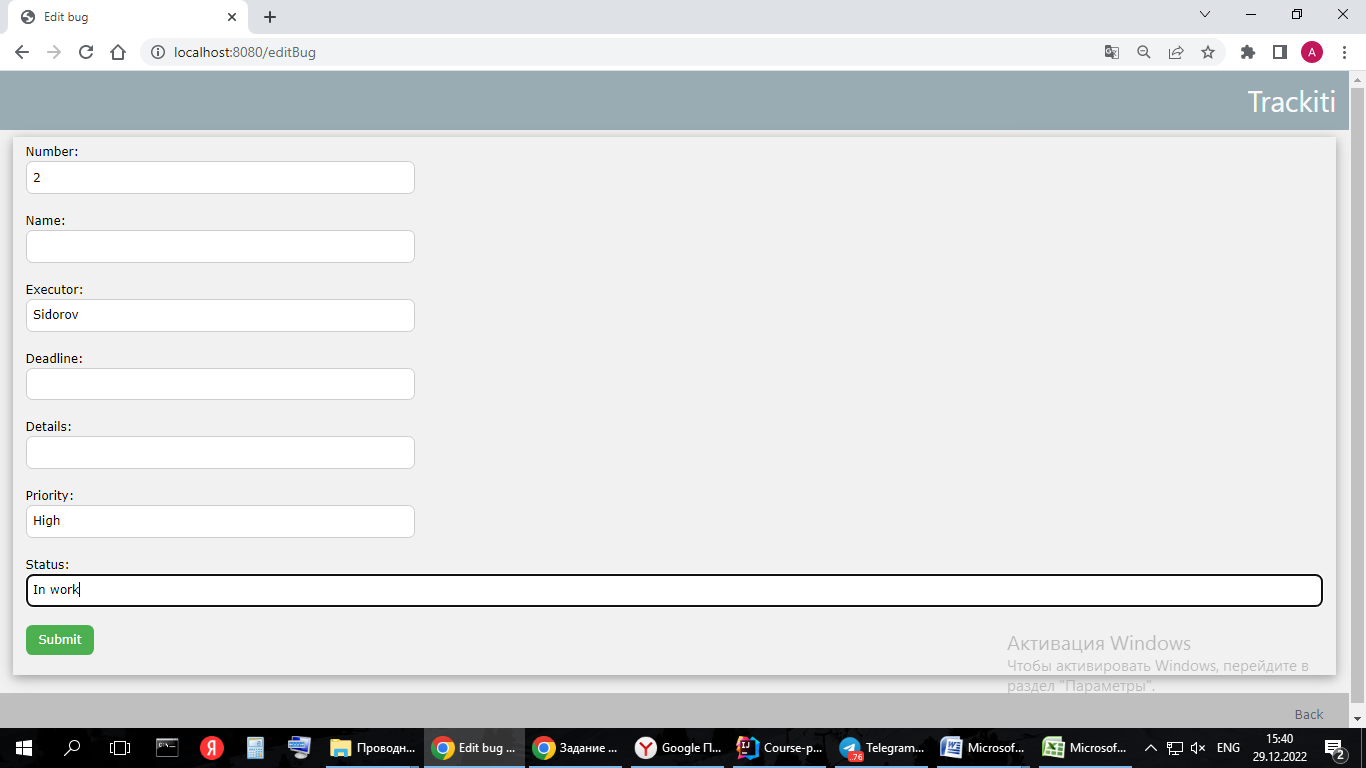


Рисунок 7.4 — Редактирование бага

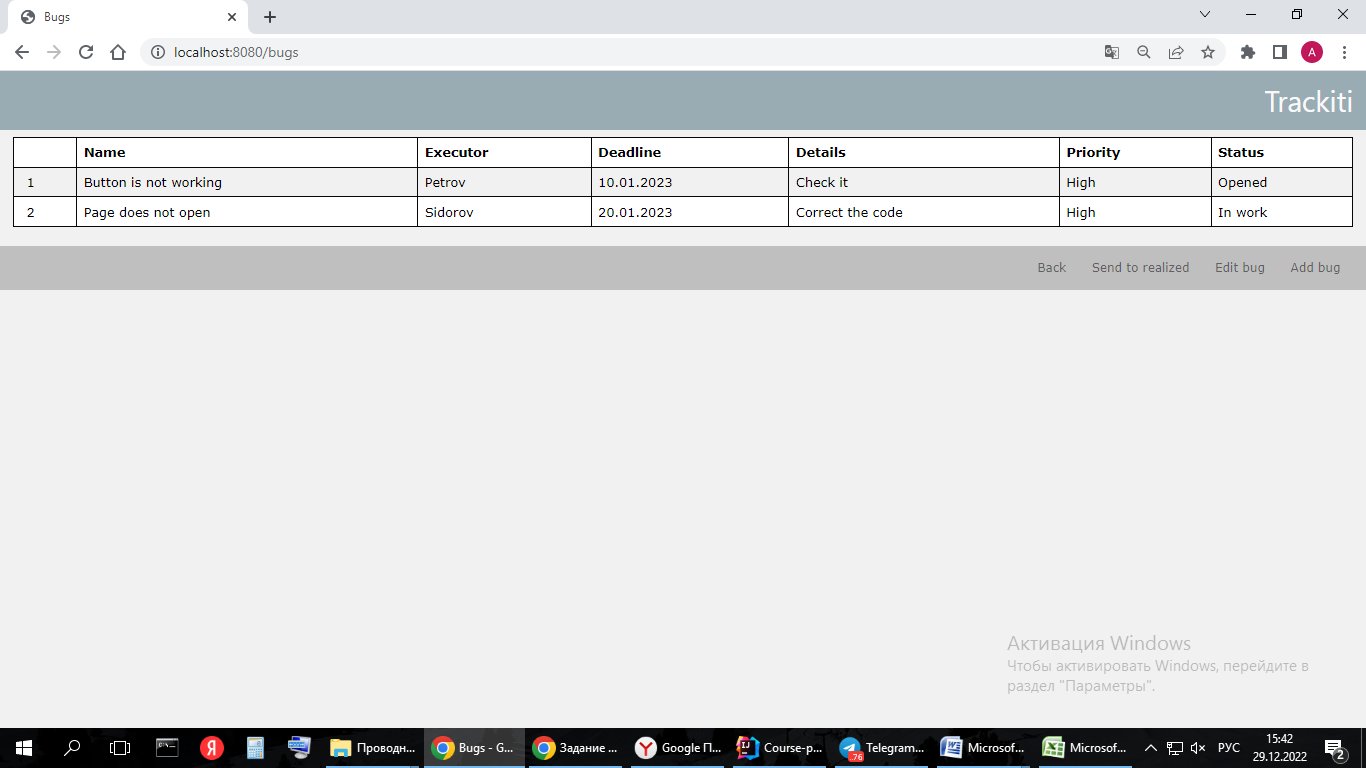


Рисунок 7.5 — Результат редактирования

С выводом всё хорошо. Теперь проведём аналогичные действия на страницах, относящихся к фичреквестам (рис.7.6 – 7.9).

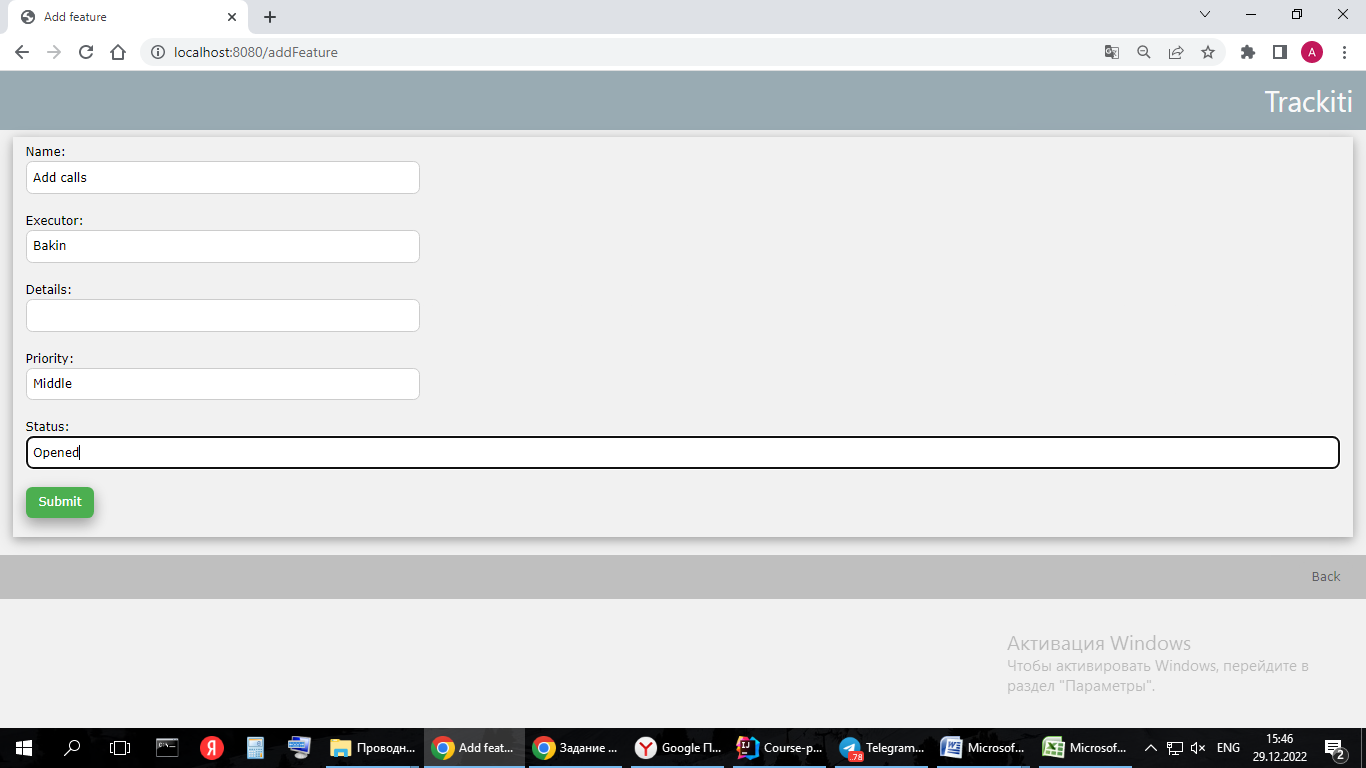


Рисунок 7.6 — Добавление фичреквеста

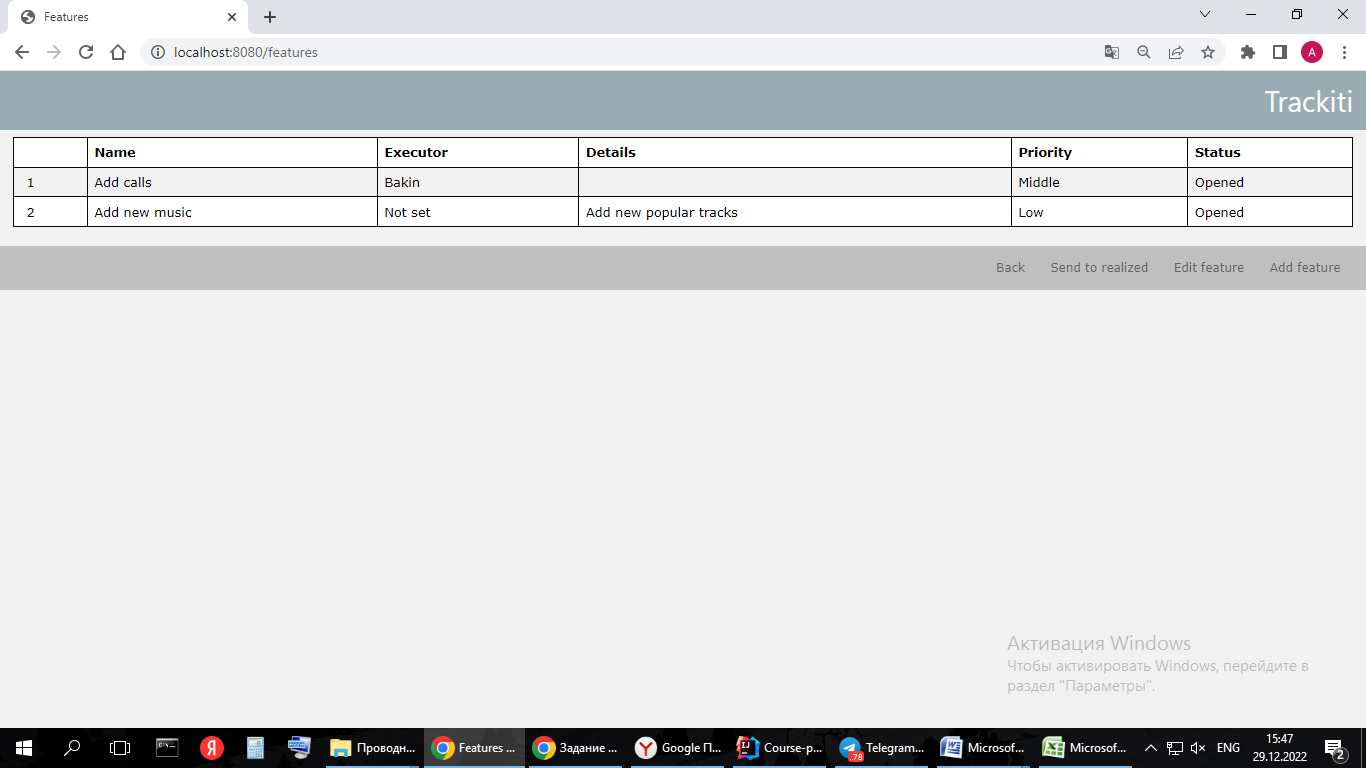


Рисунок 7.7 — Отображение результатов

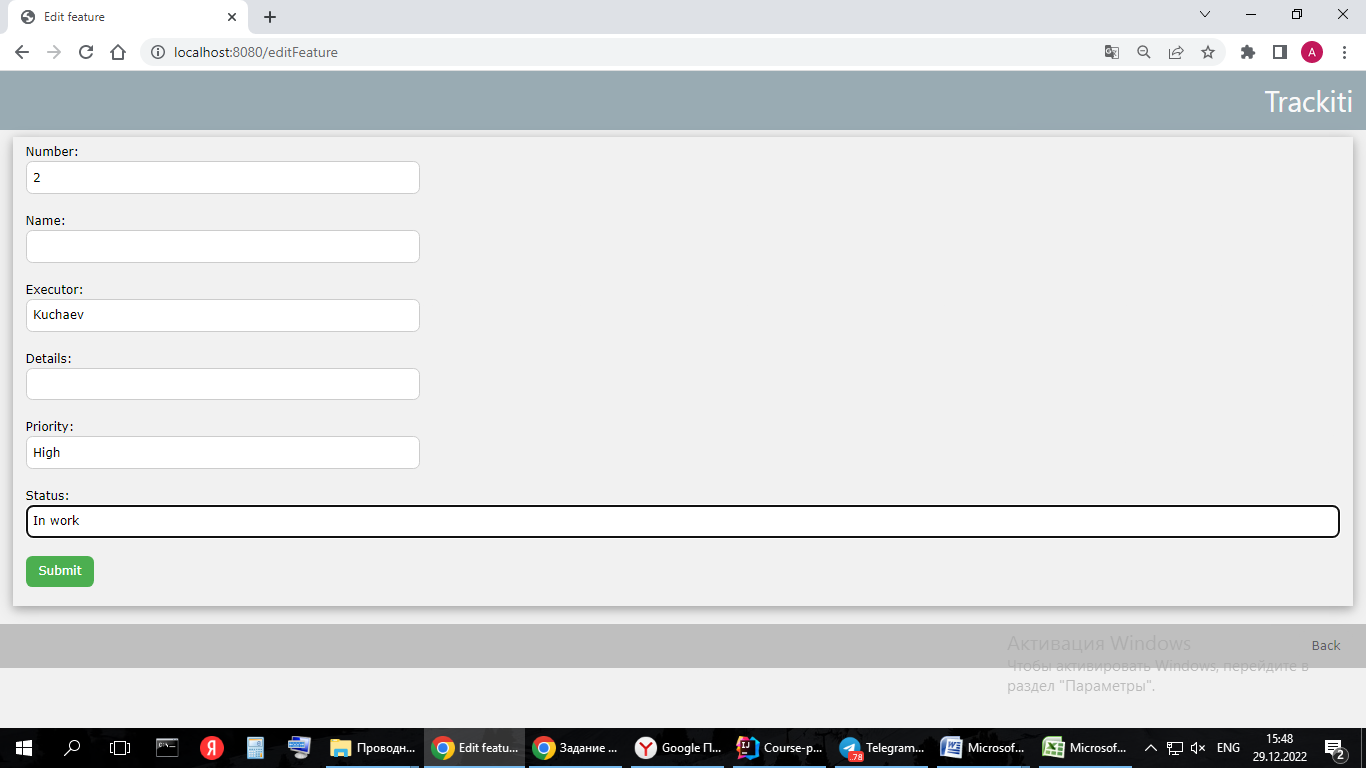


Рисунок 7.8 — Редактирование фичреквеста

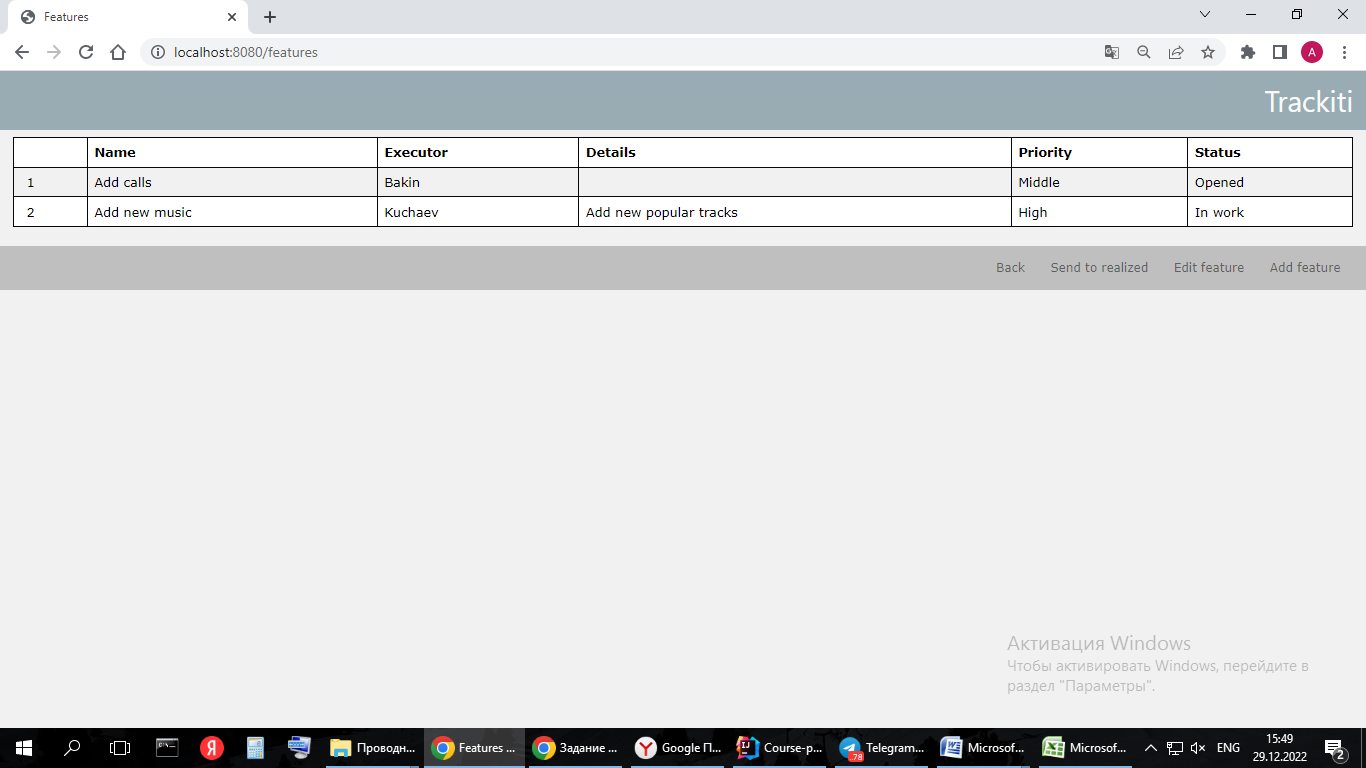


Рисунок 7.9 — Результат редактирования

Всё работает хорошо. Теперь попробуем отправить баги и фичреквесты в реализованное (рис.7.10 – 7.11) и посмотрим на результат (рис.7.12).

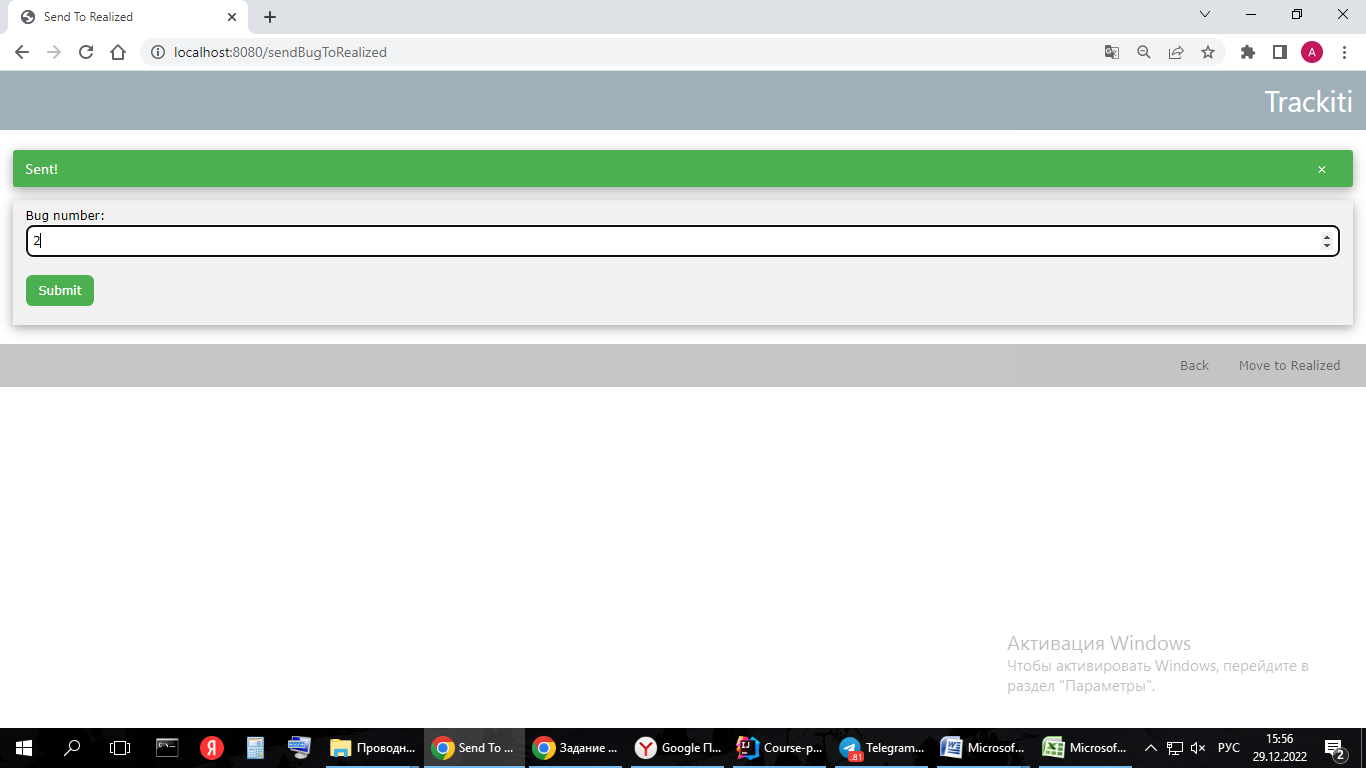


Рисунок 7.10 — Отправление бага в реализованное

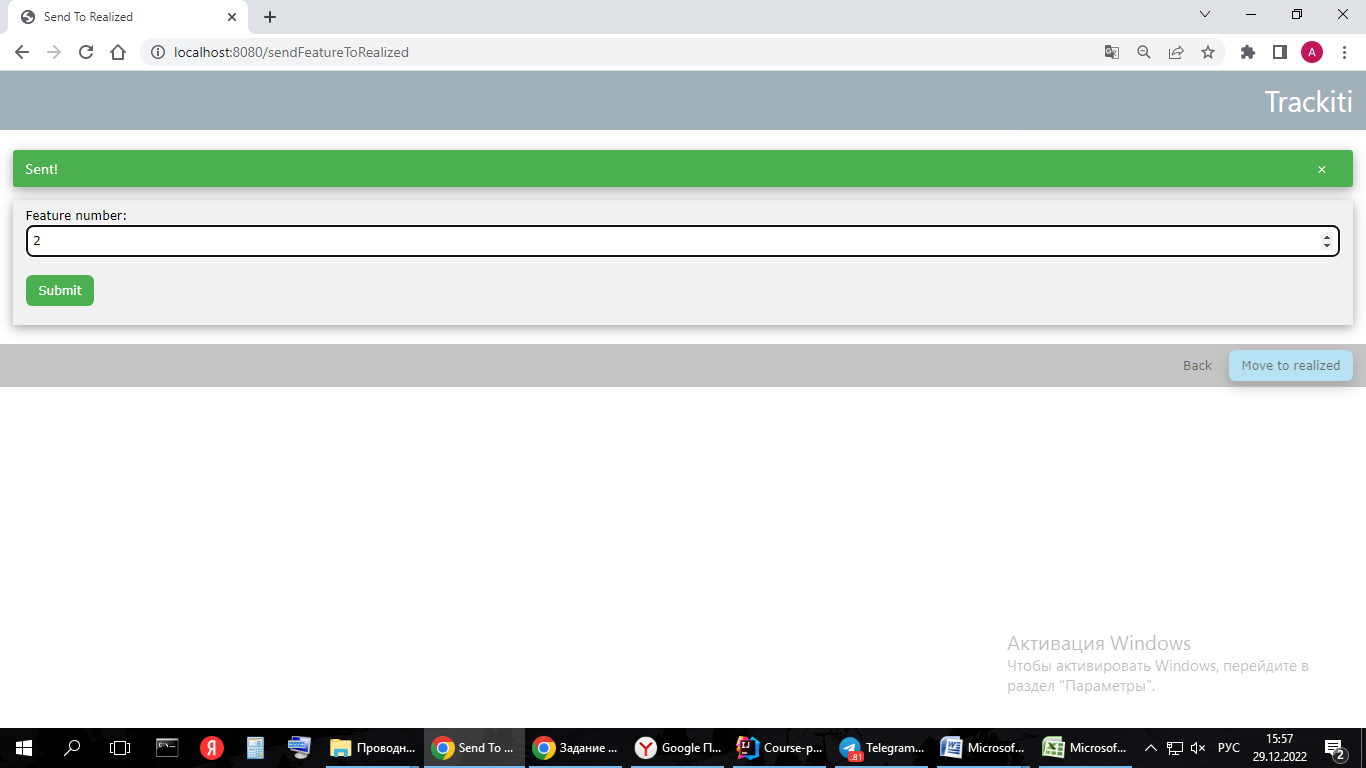


Рисунок 7.11 — Отправление фичреквеста в реализованное

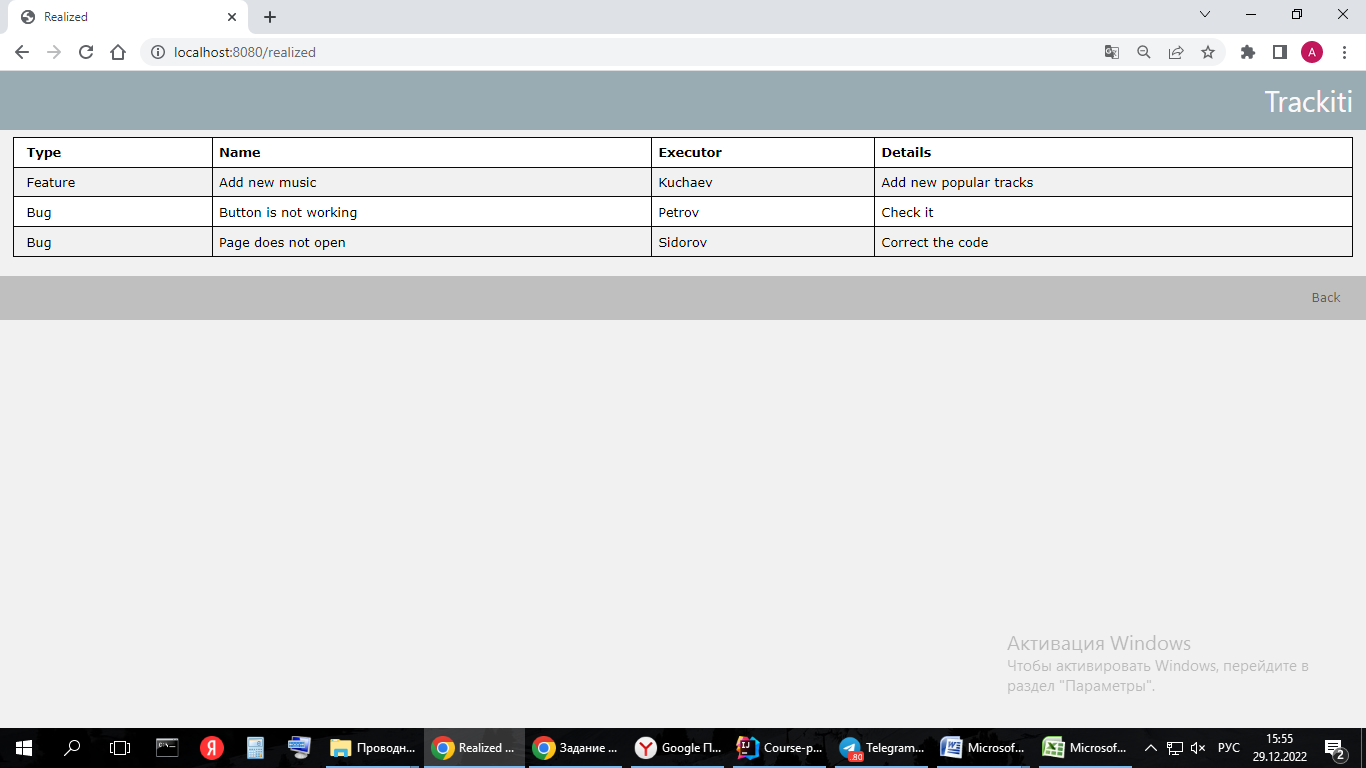


Рисунок 7.12 — Страница с реализованным

Всё работает, как и планировалось.

КОД РЕАЛИЗАЦИИ СИСТЕМЫ

Код программы и документация находятся в репозитории по ссылке: [*https://github.com/hhhandrw/Course-project-Trackiti.git*](https://github.com/hhhandrw/Course-project-Trackiti.git).

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было реализовано приложение, предназначенное для учёта багов и фиреквестов.

После написания программы были реализованы тесты приложения, которые показали работоспособность данного приложения. Также были указаны схемы приложения, применяемые технологии программирования.