ГУАП
КАФЕДРА № 51

ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук		Е. М. Линский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ

БАГ-ТРЕКЕР

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛИ

СТУДЕНТЫ ГР. №	5711		В. А. Веселова П. В. Шарко
		подпись, дата	инициалы, фамилия

Санкт-Петербург, 2019 г.

Содержание

1. Функциональная спецификация

Баг-трекер учитывает и контролирует ошибки (баги), найденные в программах, а также следить за процессом устранения этих ошибок.

Баг-трекер представлять собой веб-приложение, имеющее авторизацию пользователей. Регистрация и вход пользователей представляют собой отдельные страницы.

Главной страницей баг-трекера является таблица багов, в которой прописывается название бага и его другие характеристики. Таблица состоит из двух частей: открытые и закрытые тикеты. Если у тикета сменить статус на закрытый, то он переходит вниз списка, сверху находятся открытые тикеты. Между собой открытые/закрытые баги сортируются по дате добавления. При нажатии на ссылку в таблице на баг, можно просмотреть подробную информацию о нем. Авторизированному пользователю доступны дополнительные функции: добавление, изменения или уделение багов.

При создании бага авторизированным, возможно создать вложенный баг, указав какой баг будет являтся родителем. Родителями могут быть только открытые баги. У любого бага может быть только один родитель или не быть его и сколько угодно детей. Любой пользователь может просматривать баги, но не может ничего изменять.

Каждый баг имеет обязательные и дополнительные поля. Обязательные поля: название, форма (баг или фича), статус (открыт или закрыт), кто отвечает, кто добавил, дата добавления. Дополнительные поля: описание, родители, дети (вложенные баги).

Все поля бага могут быть изменены авторизованным пользователем. Если баг имеет вложенные баги, его статус можно изменить на закрытый, если закрыти все его вложенные баги.

Любой баг может быть удален авторизированным пользователем. Если баг имеет вложенные баги, то удаляется сам баг и все его вложенные баги.

Баг-трекер иметь поисковую строку, для удобства поиска нужного бага. Результат ищется по совпадению с полем названия бага.

2. Руководство пользователя

Переходя по ссылке в браузере http://localhost:8080/Bug, Вы попадает на начальную страницу (рисунок 1), где Вы можете выбрать «Sigupз» (зарегистрироваться), «Login» (войти), если Вы уже зарегистрированы или «Try without authorization» (продолжить без регистрации).

Нажав на «Signup», Вы попадаете на страницу регистрации (рисунок 2), где необходимо придумать свой уникальный логин и пароль. Пароль необходимо повторить. Для завершения регистрации необходимо нажать на «Signup» (рисунок 3). Вы попадаете на главную страницу с возможностями авторизированного пользователя (рисунок 11).

Нажав на «Login», Вы попадаете на страницу входа (рисунок 4), где необходимо ввести свой логин и пароль и нажать для входа на «Login». Если такой пользователь не найден или пароль введен не верно, появляется предупреждение (рисунок 5). Вы можете зарегистрироваться, нажав на ссылку «Signup», или прдолжить без авторизации, нажав на ссылку «Try without authorization». При успешном входе Вы попадаете на главную страницу с возможностями авторизированного пользователя (рисунок 11).

Нажав на «Try without authorization», Вы попадаете на главную страницу с возможностями гостя (рисунок 6). Вам доступны просмотр таблицы багов и поиск багов по названию в поисковой строке. Если Вы хотите найти найти баг, введите его предпологаемое имя в поле «Name of bug» и нажмите на «Search» (рисунок 7). Перед Вами появится таблица с багами, имя котрых совпадает или начинается на Ваш запрос (рисунок 8). Также Вы можете посмотреть информацию о конкретном баге, нажав на ссылку в таблице с ео именем. Вы попадаете на страницу информации о баге в режиме просмотра (рисунки 9, 10). Если у бага есть родитель или дети, то Вы можете посмотреть о них иформацию нажав на ссылку с их именем.

Если вы авторизированный пользователь, то на главной странице Вам доспупны все возможности гостя и дополнительные возможности.

Вы может покинуть/выйти страницу нажав на ссылку «Logout».

Вы можете добавлять новые баги, нажав на «Add bug/feature» (рисунок 13). Попадаете на страницу создания нового бага, где обязательно нужно указать его уникальное имя, форму, статус, кому адресованно. Описание бага по желанию. Дата и от кого добавятся автоматически. Для сохрания нового бага нажмите на «Save new bug».

Чтобы удалить или изменить определенный баг, нажмите на его ссылку в таблице и на странице информации о баге будут доступны «Delete» и «Change» (рисунок 14). Для удаления бага нажмите на «Delete». Если у бага есть вложенные баги, то они будут также удаленны вместе с багом (рисунок 15). Для изменения информации о бага нажмите на «Change», Вы попадете на страницу изменения бага, где можно будет изменить все его поля (рисунок 16). Для сохранения изменений нажмите «Save».

3. Описание архитектуры программы

Используемые инструменты:

1) Серверная часть на основе сервлетов;

2) Web-сервер: Tomcat;

3) База данных: PostgreSQL;

4) Языки front-end: HTML, CSS, JavaScript;

5) Язык back-end: Java;

6) Тесты: Junit;

7) Сборщик: Maven.

Для сборки проекта используется Maven. Фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM. Для запуска веб-приложения используется Apache Tomcat, он служит веб-сервером. Servlet API является основой для технологий Java, касающихся Web и дает возможность динамически генерировать любой веб-контент, используя разные библиотеки, доступные в Java.

С помощью Maven осуществляется деплой на Apache Tomcat, что позволяет сразу же после сборки проекта, открыть его на localhost:8080. Помимо деплоя на Tomcat, Maven позволяет провести интеграционное тестирование, после которого, в случае удачного прохождения, будет открыт localhost, в случае же неудачного прохождения тестов, проект не соберётся.

Тестирование модели осуществляется с помощью автоматизированных тестов. Для этого используется — JUnit библиотека для модульного тестирования программного обеспечения на языке Java. Тестирование позволяет проверить на корректность отдельные модули исходного кода программы, а именно классы, которые составляют модель (Class ConnectonBD, Class BDUsers, Class BDBugs). В тетирование используются методы помеченые аннотациями @Before и @Afte, которые выполняются до и после запуска очередного теста, и @BeforeClass и @AfteClass, которые выполняются один раз до и после выполнения всех тестов. Метод, помеченный аннотацией @BeforeClass выполнят подключение к базе данных и создает объект класса, а помеченный @AfterClass — закрывает подключение к базе данных. Покрытие тестами классов модели составляет около 70%.

Controller (контроллер) интерпретирует действия пользователя, оповещая модель о необходимости изменений. Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия. Контроллер предстален классами HttpServlet (классы регистрации, входа, выхода; классы создания/изменения/удаления бага, информации о баге, класс таблицы багов), использующие методы doPost для «скрытой» передачи данных.

Model (модель) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.В модель входят три класса по работе с базами данных: ConnectonBD, BDUsers, BDBugs. Модель работает с подключенной базой данных PostgreSQL. Модель отправляет запросы в базу данных, проверяет их на корректность. Реализация model представлена на языке Java.

View (представление) отвечает за отображение данных модели пользователю, реагируя на изменения модели. После того, как модель передала изменения, происходит отображение пользователю новых данных. В проекте представление реализуется jsp- и html-страницами. Основными страницами проекта являются: страницы

регистрации и входа, страница с таблица багов, создание/изменение бага и страница информации о баге. Также используются алерты, всплывающие уведомления для пользователя. Единый стиль оформления страниц задан в css-файлах.

На рисунке 17 представлено взаимодействие Model-View-Controller.

4. Описание наиболее важных классов программы

4.1. Model-классы

1) Class ConnectionBD — подключениие к базе данных PostgreSQL:
   - void connectionUsers() — подключение к базе данных пользователей.
   - void connectionBugs() — подключение к базе данных багов.
   - void close() — закрытие всех подключений.
2) Class DBBugs — взаимнодействие с базой данных багов:
   - void setBug (String login, String password) — создание нового бага и запись его данных в базу данных.
   - ArrayLis getBug (String name) — получение всех данных о баге по именю.
   - ArrayLis getBugs () — получение всех багов из базы данных.
   - void deleteBug (String name) — удаление бага по именю.
   - void changeBug (String name, String status, String newNname,String fromwho, String towho, String data, String info, String bf, String parent) — изменение данных о баге по имени.
   - ArrayList newListBugDelete (String nameBug) — возражает список вложенных багов, которые необходимо удалить. Вызывается при удалении бага, у которого есть вложенные баги. В основе реализации лежит создание очереди, при прохождении по базе данных.
   - ArrayList newListBugParent (String nameBug) — возражает список багов, которые потенциально могут быть родителями. Вызывается при измении бага. В основе реализации лежит создание очереди, при прохождении по базе данных.
   - boolean getStatus (String name) — возращает статус бага.
3) Class BDUsers — взаимнодействие с базой данных пользователей:
   - void setUser (String login, String password) — создание нового пользователя и запись его данных в базу данных.
   - boolean getUser (String login) — проверка наличия пользователя в базе данных.

- boolean authentication (String login, String password) — аутентификация пользователя в базе данных.
- ArrayList getUsers() — возращает список всех пользователей.

4.2. Tests-классы

1) Class ConnectionBDTest — тестирование класса ConnectionBD:
   - void testCheckConnection — тестирует произошло ли подключение подключение к базам данных.
   - void setUp() — имеет аннотацию @BeforeClass, создает экземпляр класса ConnectionBD.
   - void close() — имеет аннотацию @AfterClass, закрывает все подключения к базам данных.
2) Class DBBugsTest —тестирование класса DBBugs:
   - void setUp () — имеет аннотацию @Before, создание экземпляра класса BDBugs и списка с информацией о баге.
   - void close() — имеет аннотацию @After, закрывает подключения к базе данных багов.
   - void testGetBug () — теститрует правильность полученнных данных о баге.
   - void testGetBugs () — тестирует правильность полученного списка багов из базы данных.
   - void testNewListBugDelete () — тестирует правильность списока вложенных багов, которые необходимо удалить.
   - void testNewListBugParent () —тестирует правильность списока багов, которые потенциально могут быть родителями.
   - void testGetStatus () — тестирует правильность полученнго статуса бага.
3) Class BDUsersTest — тестирование класса BDUsers:
   - void setUp () — имеет аннотацию @Before, создание экземпляра класса BDUsers и списка всех пользователей.
   - void close() — имеет аннотацию @After, закрывает подключения к базе данных пользвателей.
   - voig testLogin() — тестирует проверку наличия пользователя в базе данных.
   - void testAuthentication — тестирует аутентификация пользователя в базе данных.
   - void testGetUsers() — тестирует полученные данные (список имеющихся пользователей) базы данных.

5. Области выполненной работы

Веселова Виктория	Шарко Полина
Разработка серверной части Разработка поисковой системы багов Разработка сортировки багов Написание отчета	Разработка клиентской части Разработка интерфейса Подключение и создание баз данных Тестирование

6. Ссылки

[1] Проект «Bug-Tracker» на Github.

7. Приложение

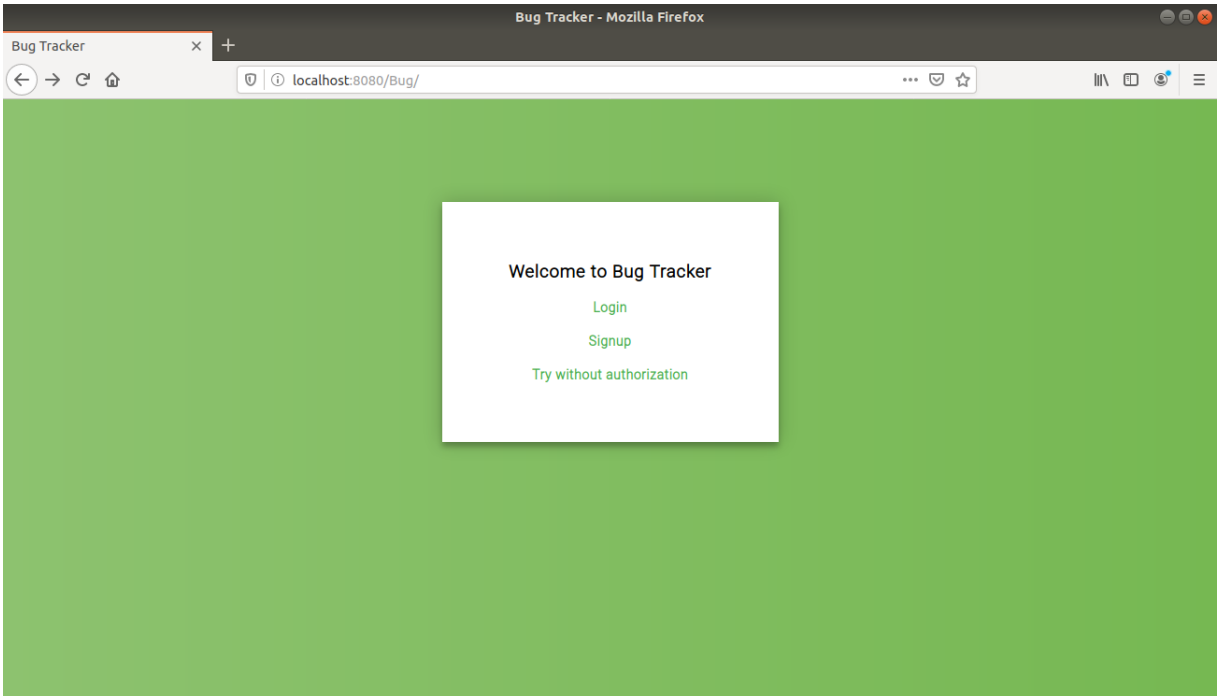

Рис. 1. Начальная страница баг-трекера.

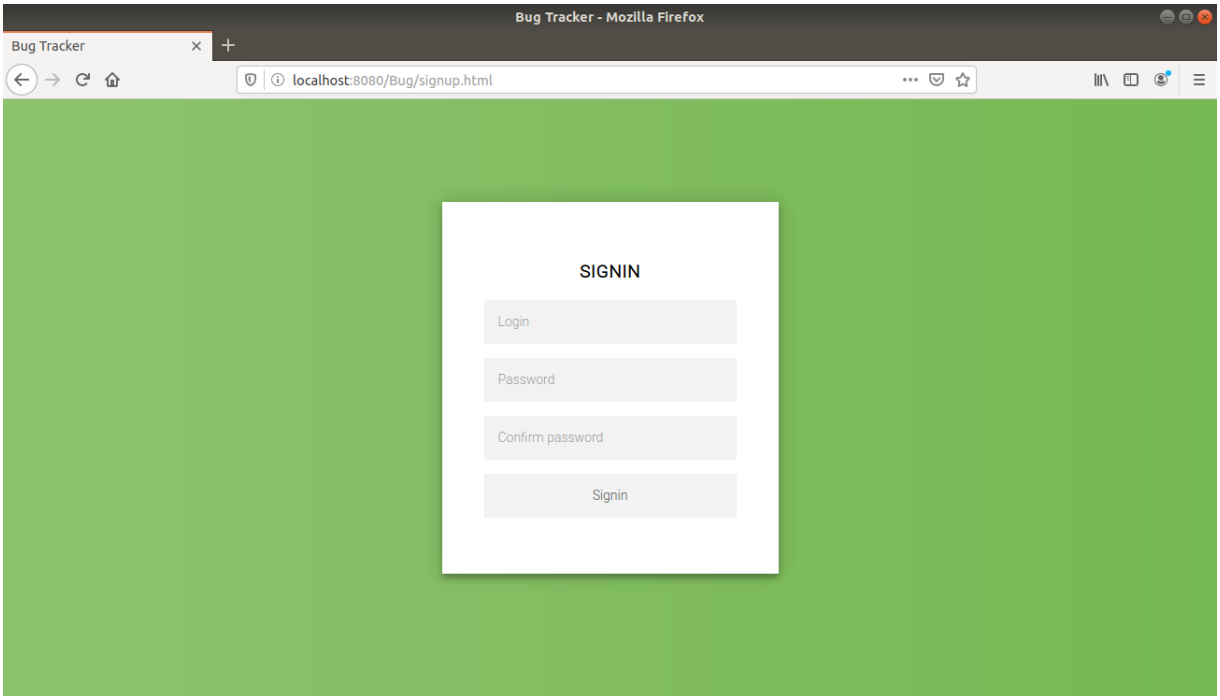

Рис. 2. Регистрация пользователя

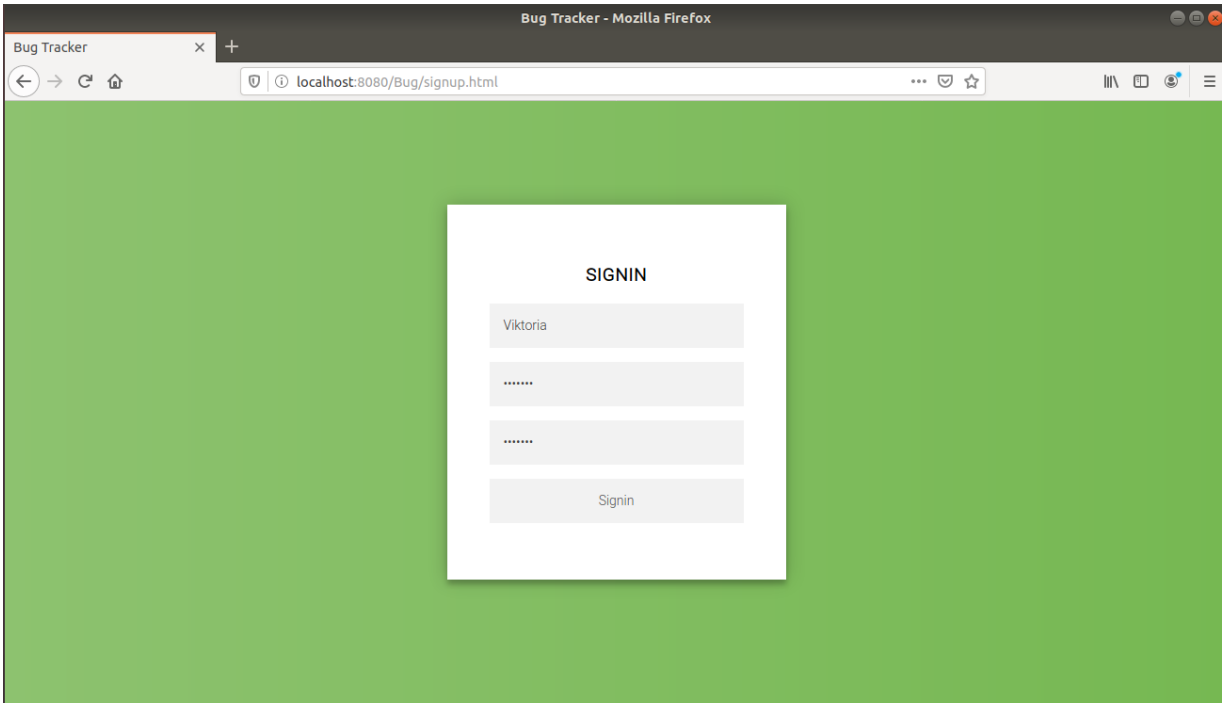

Рис. 3. Ввод данных при регистрации

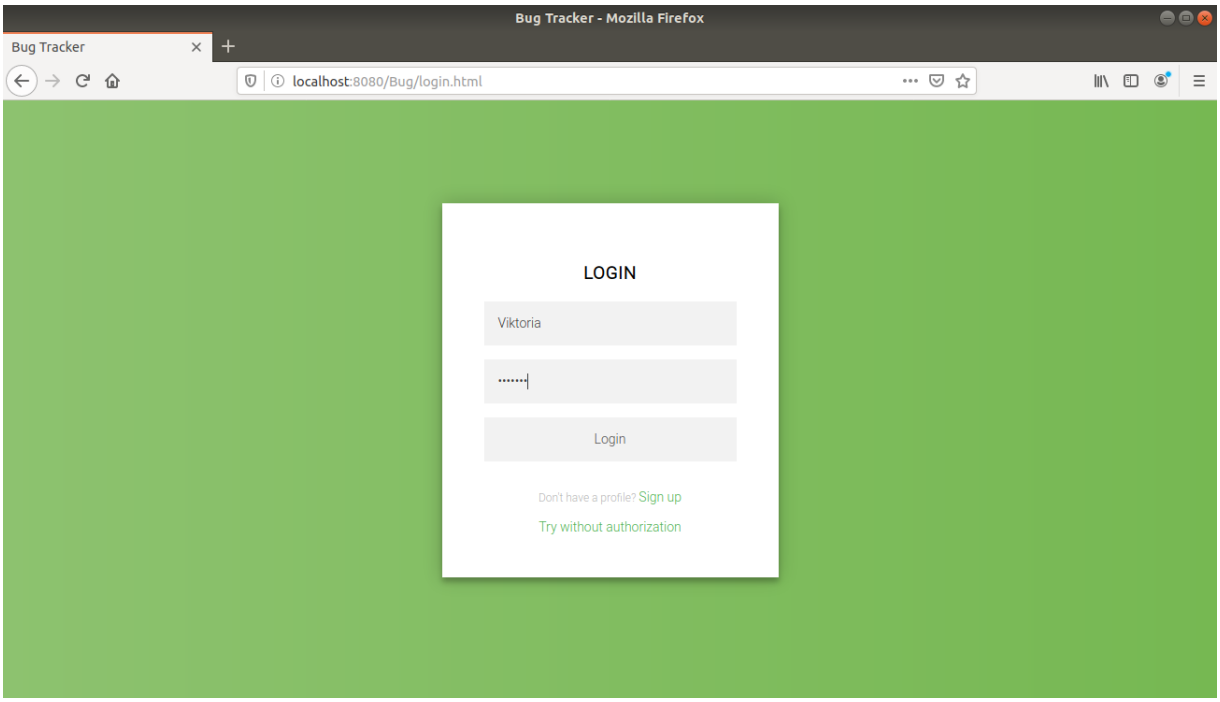

Рис. 4. Вход пользователя

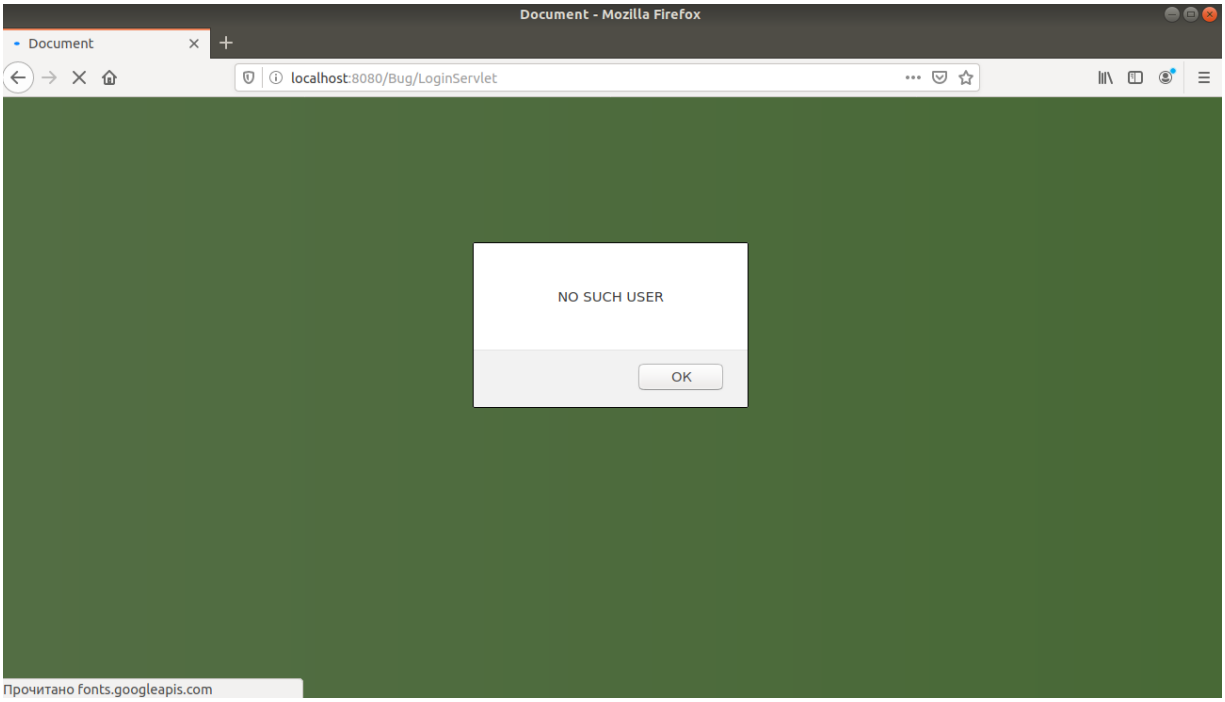

Рис. 5. Неверная авторизация

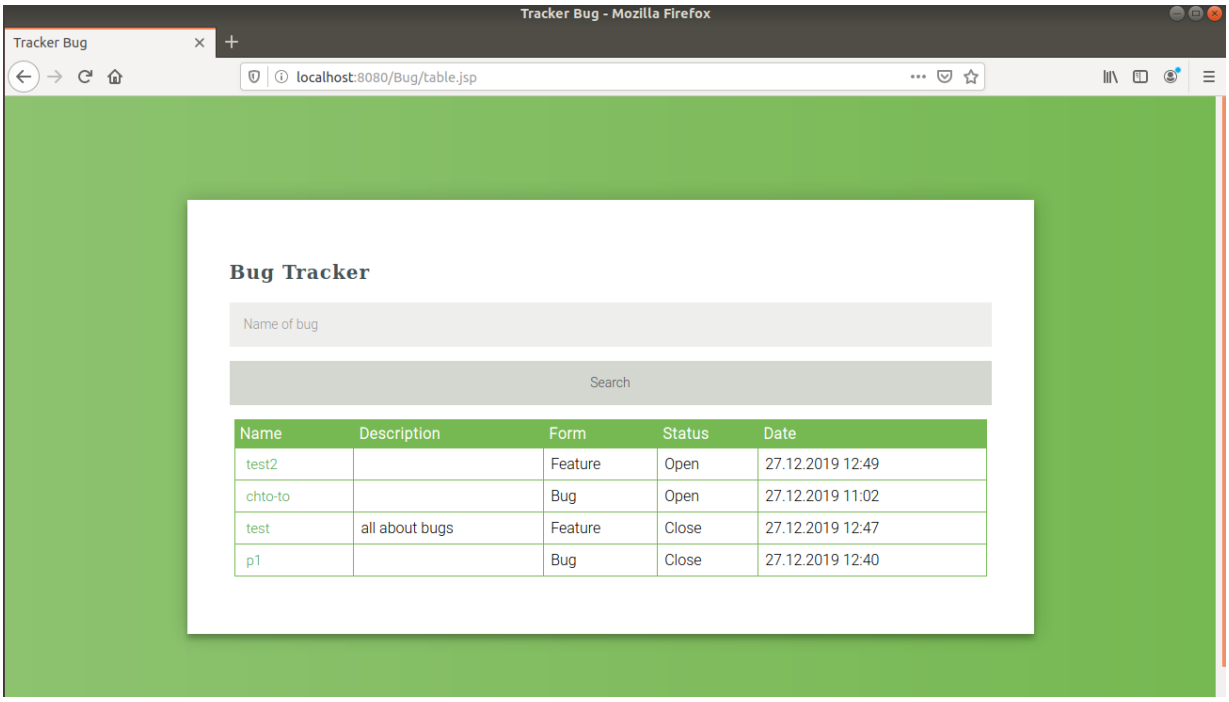

Рис. 6. Главная страница баг трекера в гостевом режиме

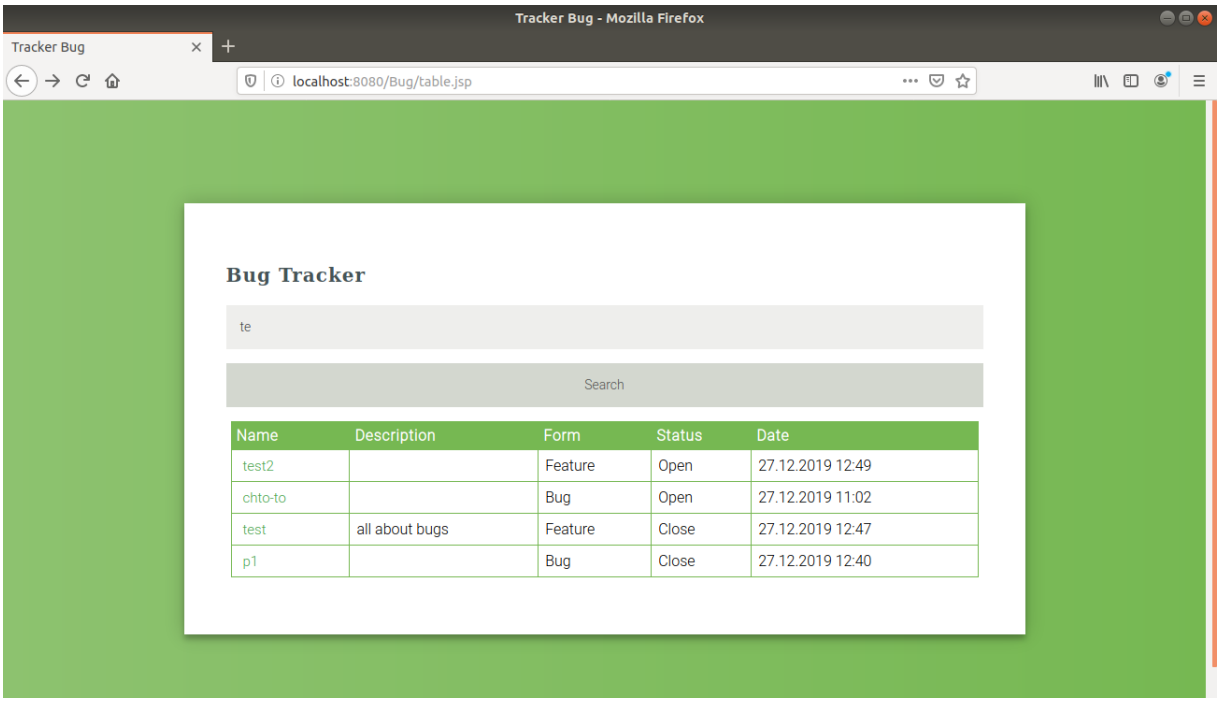

Рис. 7. Поисковой запрос

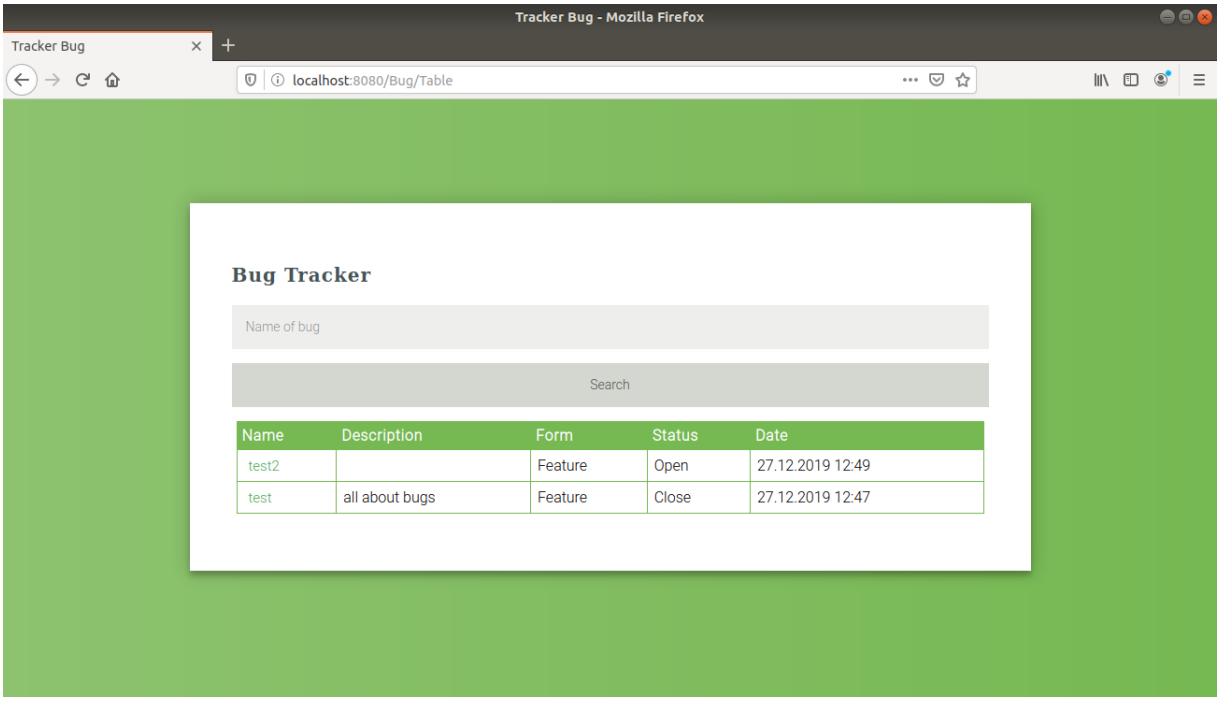

Рис. 8. Результаты поиска

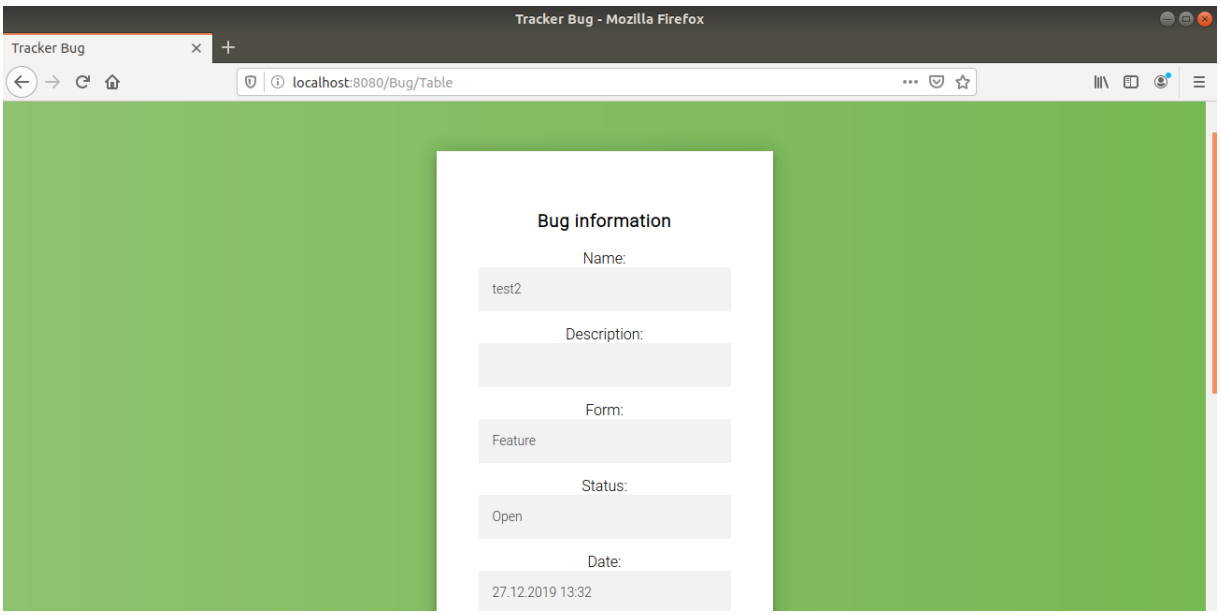

Рис. 9. Информация о баге в гостевом режиме

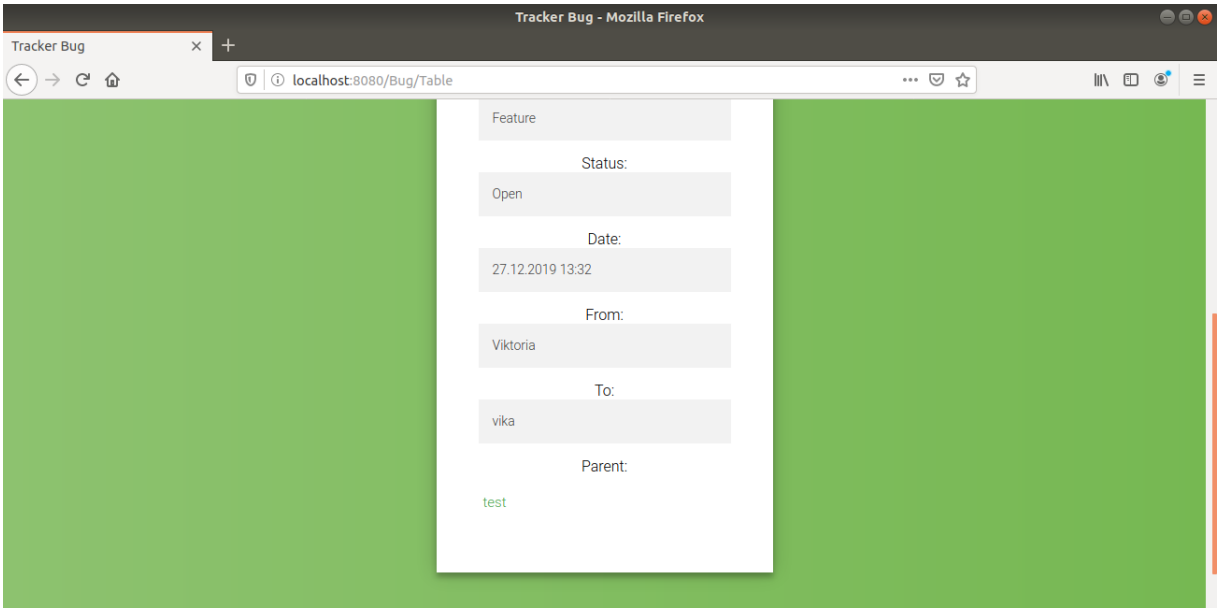

Рис. 10. Информация о баге в гостевом режиме

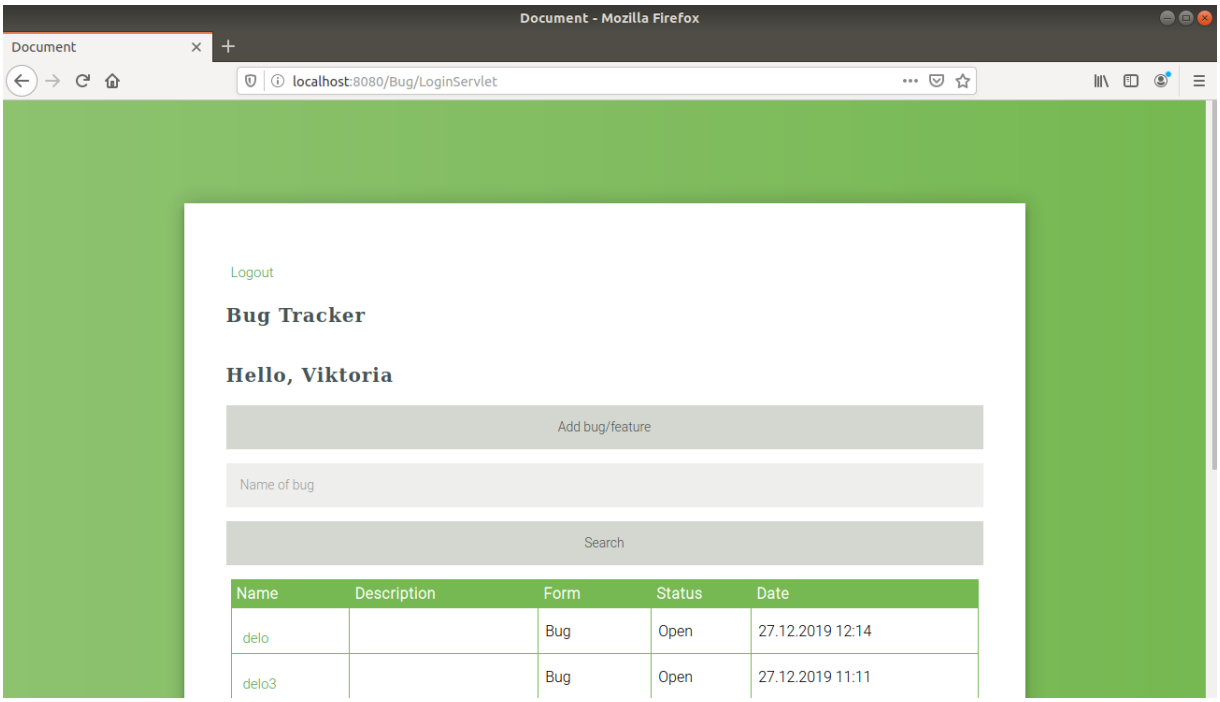

Рис. 11. Главная страница баг-трекера в режиме авторизированного пользователя

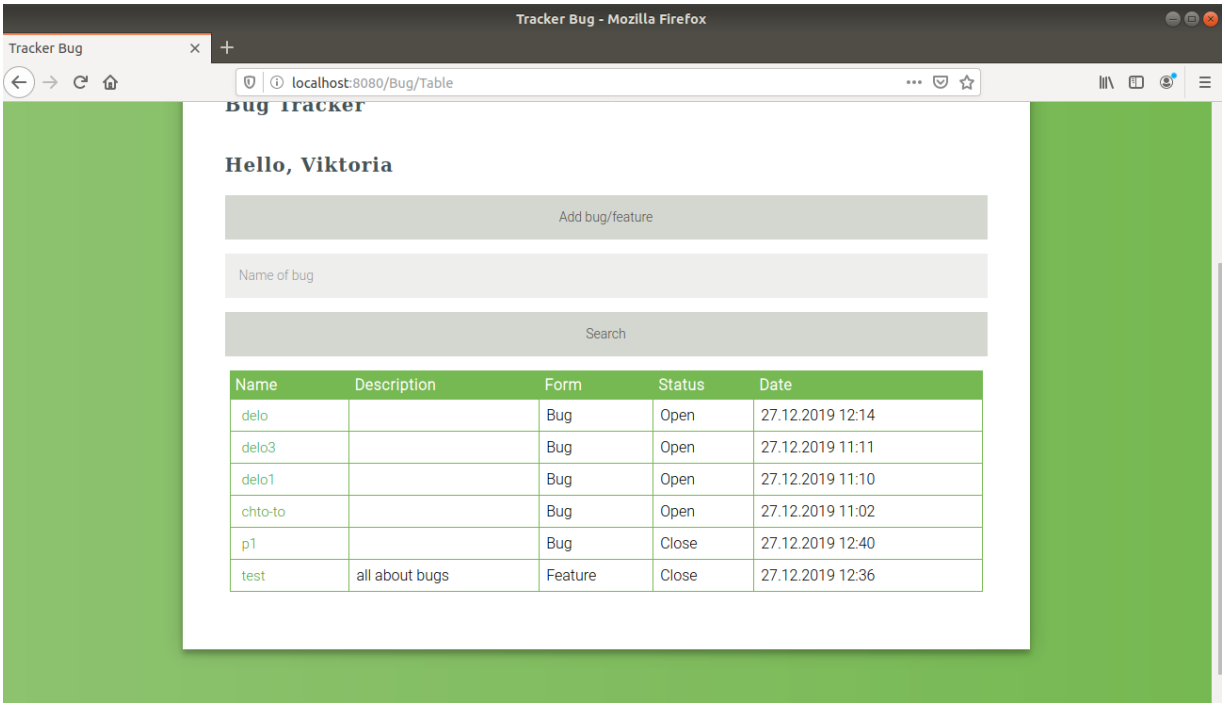

Рис. 12. Таблица багов

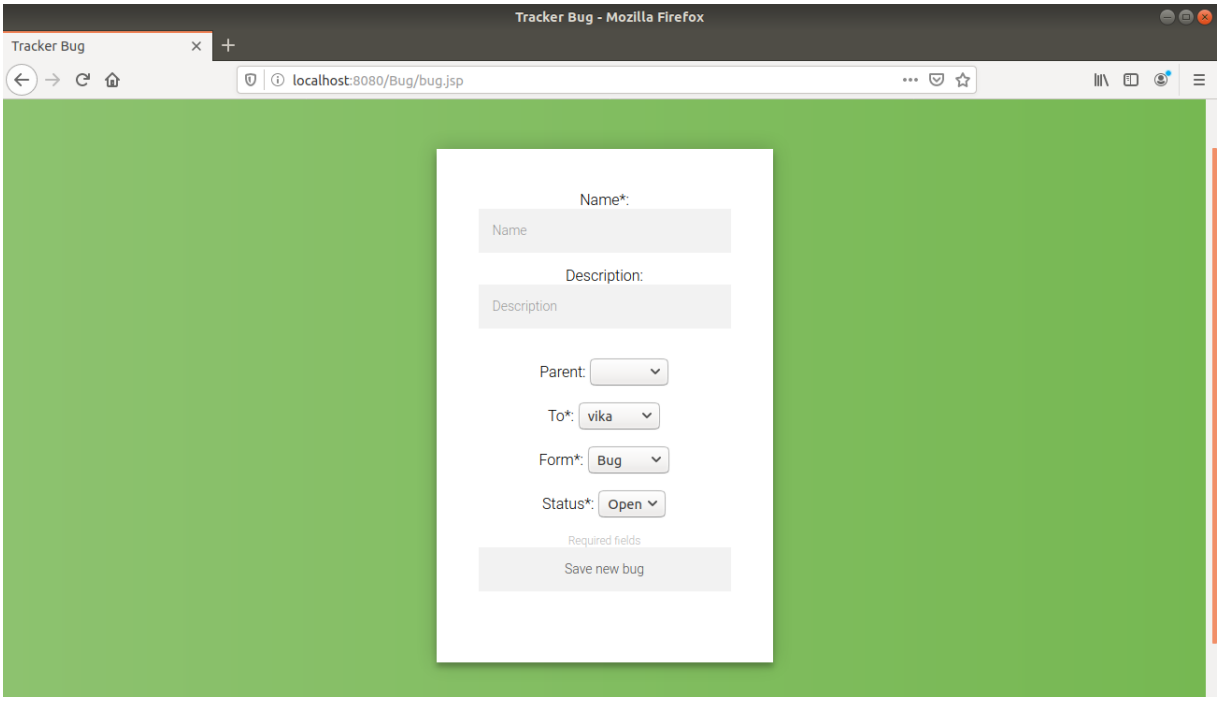

Рис. 13. Создание нового бага

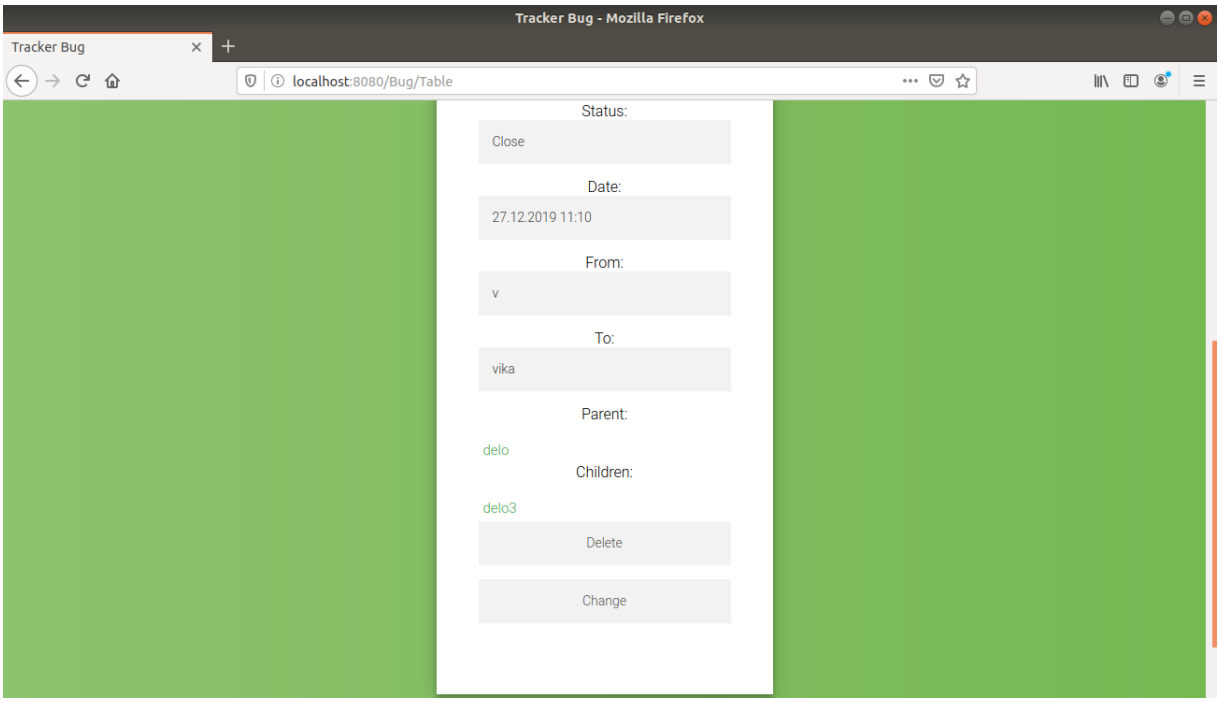

Рис. 14. Информация о баге в режиме авторизированного пользователя

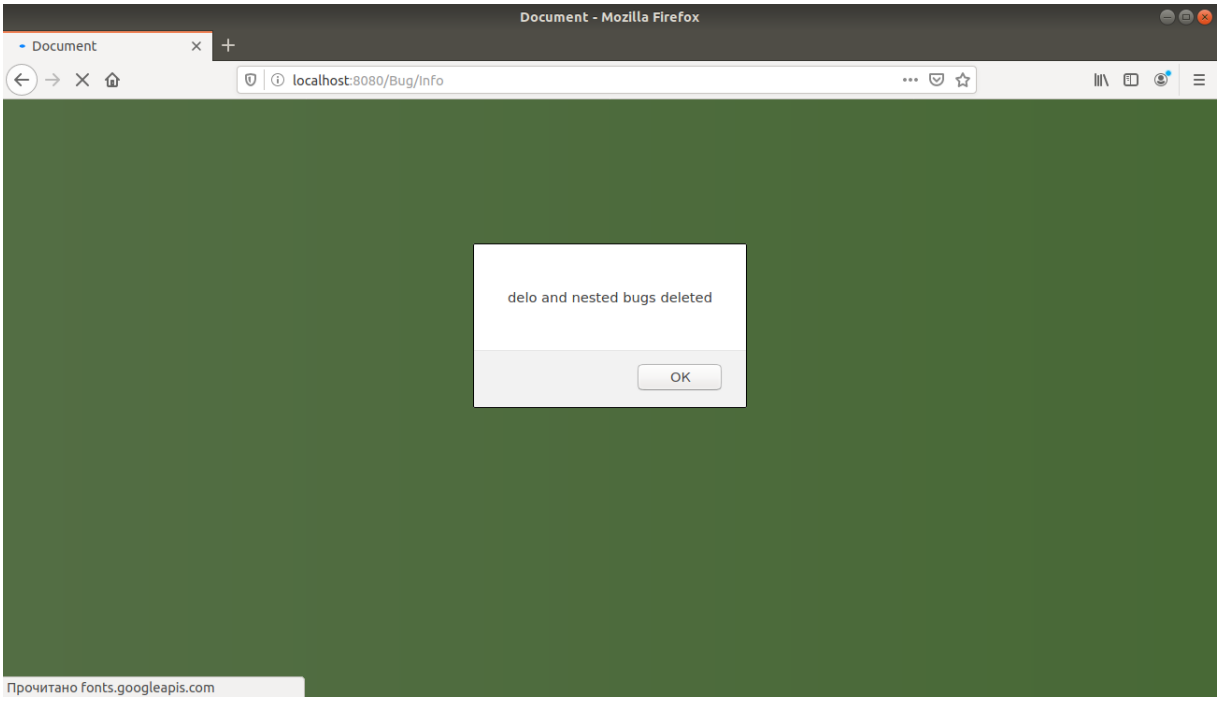

Рис. 15. Удаление бага

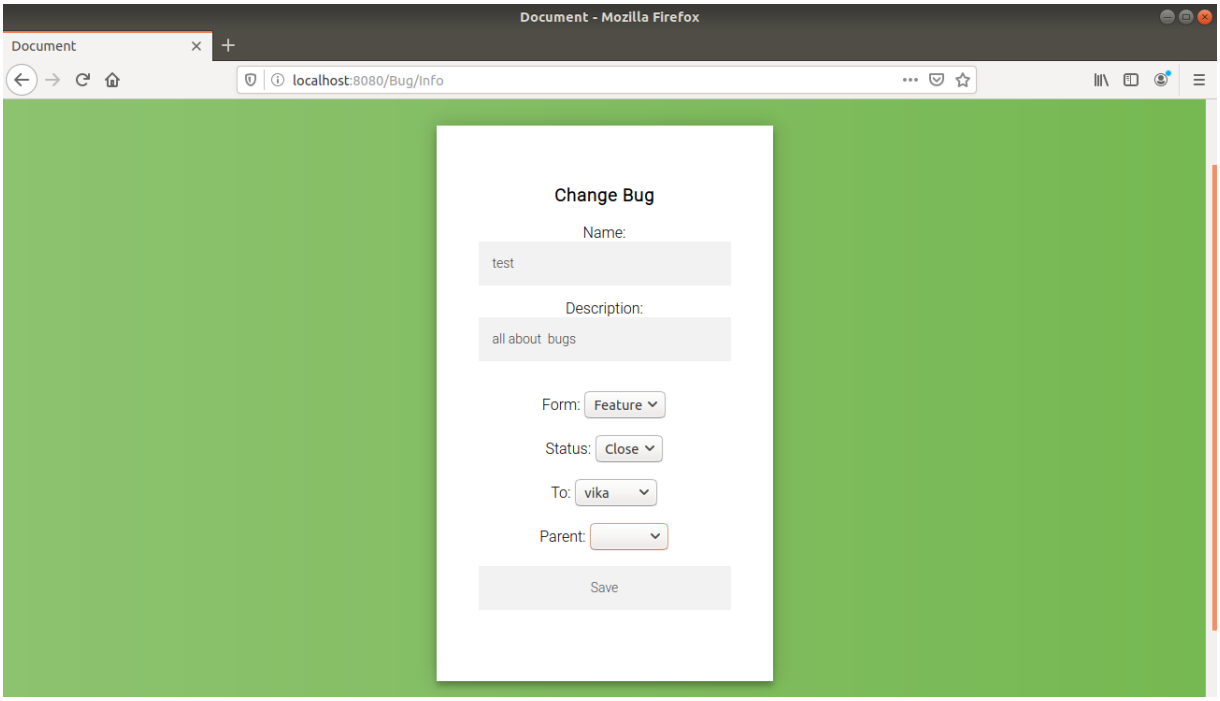

Рис. 16. Изменение бага

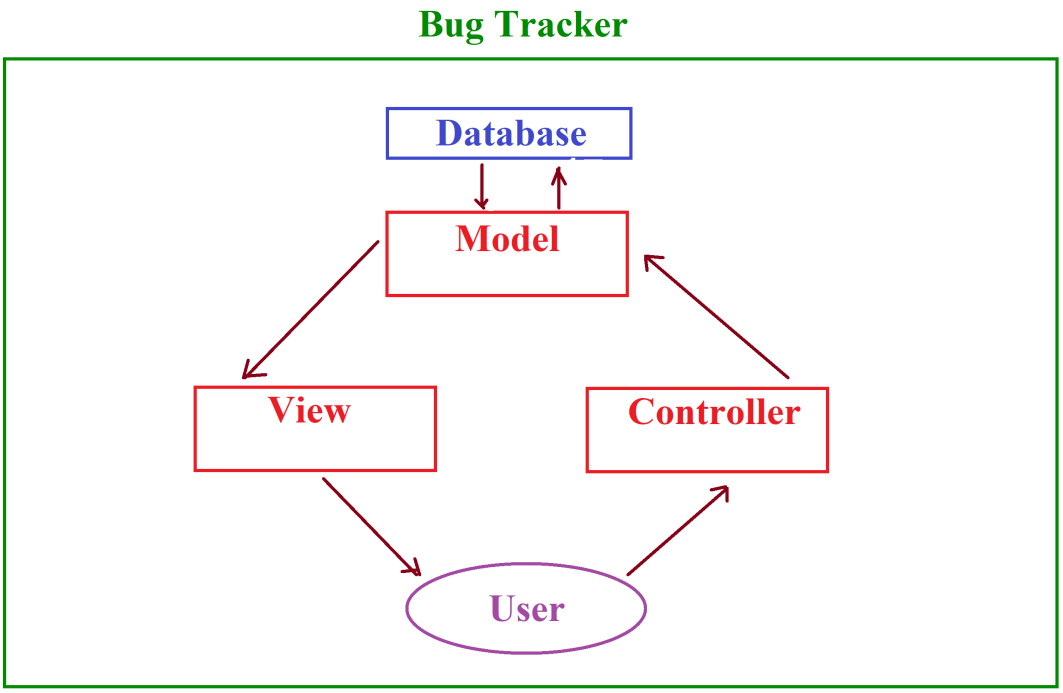

Рис. 17. Графическая схема архитектуры программы