Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСиС»
Институт ИТКН Кафедра АСУ

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

«Поддержка и развитие сервиса GitFlic»

Выполнил:
ст. гр. БИВТ-20-3
Меньшов Алексей Сергеевич

Проверил:
к.т.н., доцент кафедры АСУ
Громов Сергей Владимирович

Москва 2022

Оглавление

Введение **2**

1. Цели и задачи практики **3**

2. Описание предметной области **4**
2.1 Описание работы Git 4
2.2 Описание сервиса хранения исходного кода GitFlic 5

3. Реализация функционала уведомлениях о событиях в проектах GitFlic в Telegram. **7**
3.1 Технологический стек 7
3.1.2 Spring Framework 8
3.1.3 Spring Boot 9
3.1.4 Hibernate 9
3.1.5 Spring Data Jpa 9
3.2 Telegram Bot API. 10
3.3 Интеграция GitFlic с Telegram Bot API 11
3.3.1 Постановка задачи 11
3.3.2 Алгоритм интеграции 12
3.3.3 Реализация 13
3.3.4 Отправка сообщения 15

Заключение **17**

Список литературы **18**

Введение

GitFlic - первый российский сервис для хранения кода и работы с ним, который основан на системе контроля версий Git.

С помощью российского облачного сервиса хранения репозиториев исходного кода отечественные вузы и компании, которые находятся под санкциями, найдут поставщика услуг для размещения исходного кода своих технологических решений. При этом данные будут храниться в расположенных на территории РФ сертифицированных дата-центрах.

Данная платформа позволяет хранить исходных код приложения, а также предоставляет различных инструментарий для работы в команде, такой как создание проблем (issue), запросов на слияние (merge request), настройка прав доступа к репозиторию и прочее.

1. Цели и задачи практики

Цель практики: изучить работу сервиса, а также осуществлять его поддержку путем исправления программных ошибок, а также разработкой новых задач и улучшений сервиса.

Задачи:

- Изучение системы работы Git;
- Изучение системы работы GitFlic;
- Реализовать функционал уведомления о событиях в проекте в мессенджер Telegram.

2. Описание предметной области

2.1 Описание работы Git

Git — система управления версиями с распределенной архитектурой. В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием. Это позволяет всем разработчикам хранить историю изменений в полном объеме.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы. Структура

хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов.Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

2.2 Описание сервиса хранения исходного кода GitFlic

GitFlic является сервисом для хранения исходного кода и работы с ним, основанный на системе контроля версий Git.

Отличительными функциями сервиса являются:

- Возможность работать в команде - создать свою команду или присоединиться к готовой. Команду можно сделать как приватной, так и публичной.
- Запросы на слияние – возможность создать запросы на слияние веток и обсудить изменение кода с командой или с друзьями. Все находится под контролем и с гибкими настройками прав.
- 2fa защита профиля – для дополнительной защиты аккаунта от взлома.
- Наличие и публичных, и приватных репозиториев – для настройки доступа к рабочему проекту.
- Обсуждение кода - возможность комментировать участки кода и обсуждать их изменение.

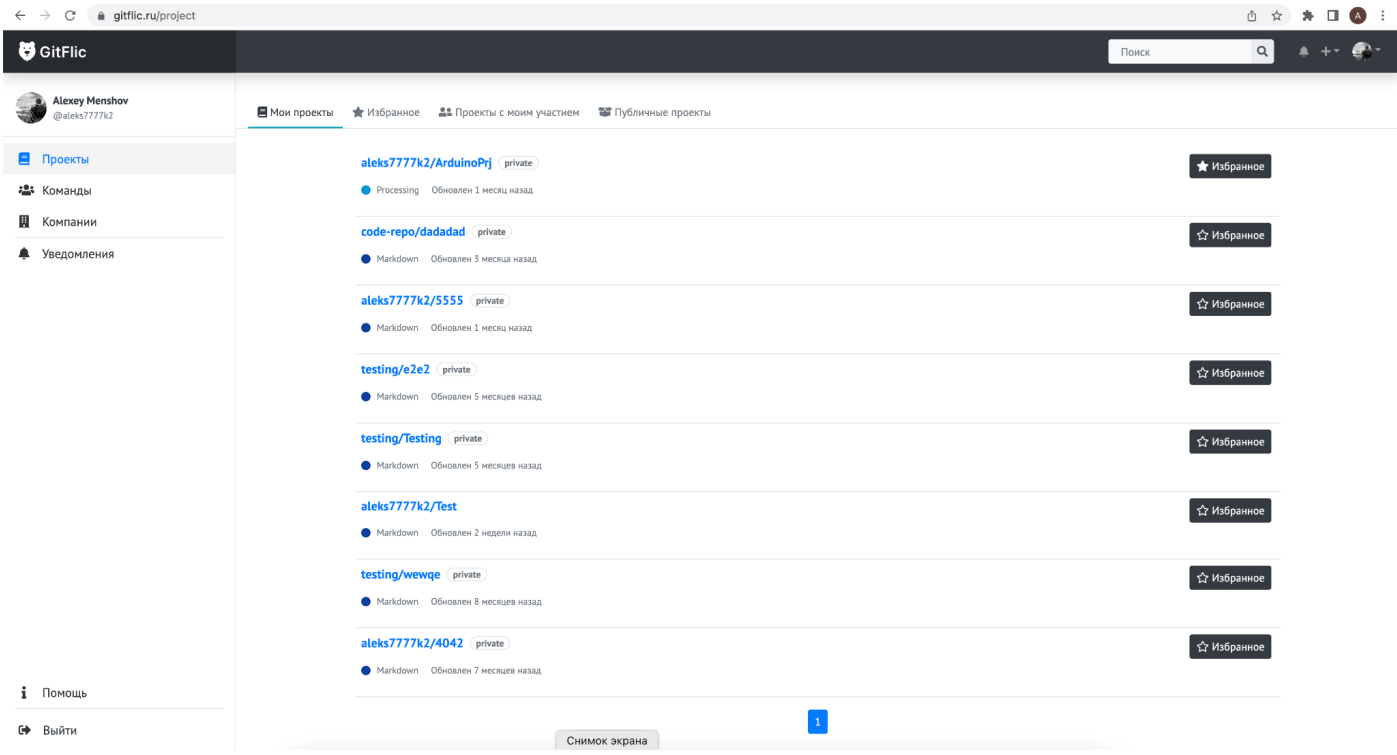

Рисунок 1 – страница личных проектов

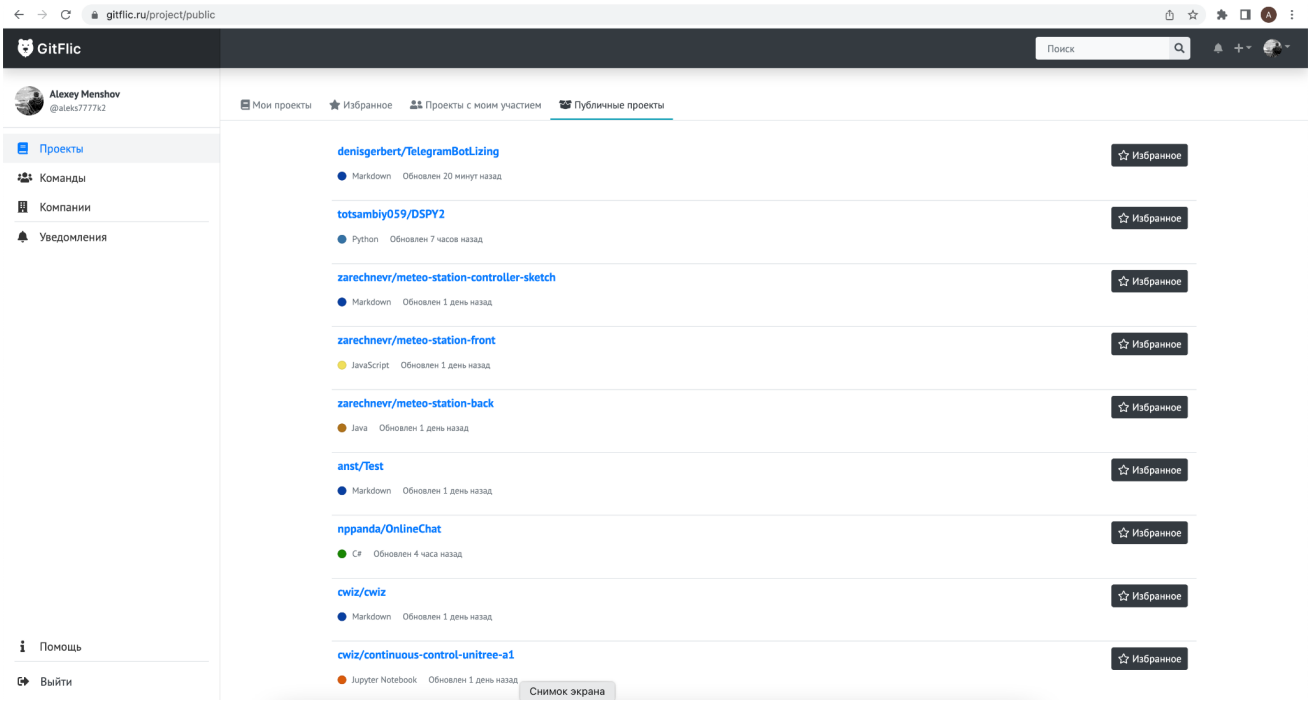

Рисунок 2 – страница личных проектов

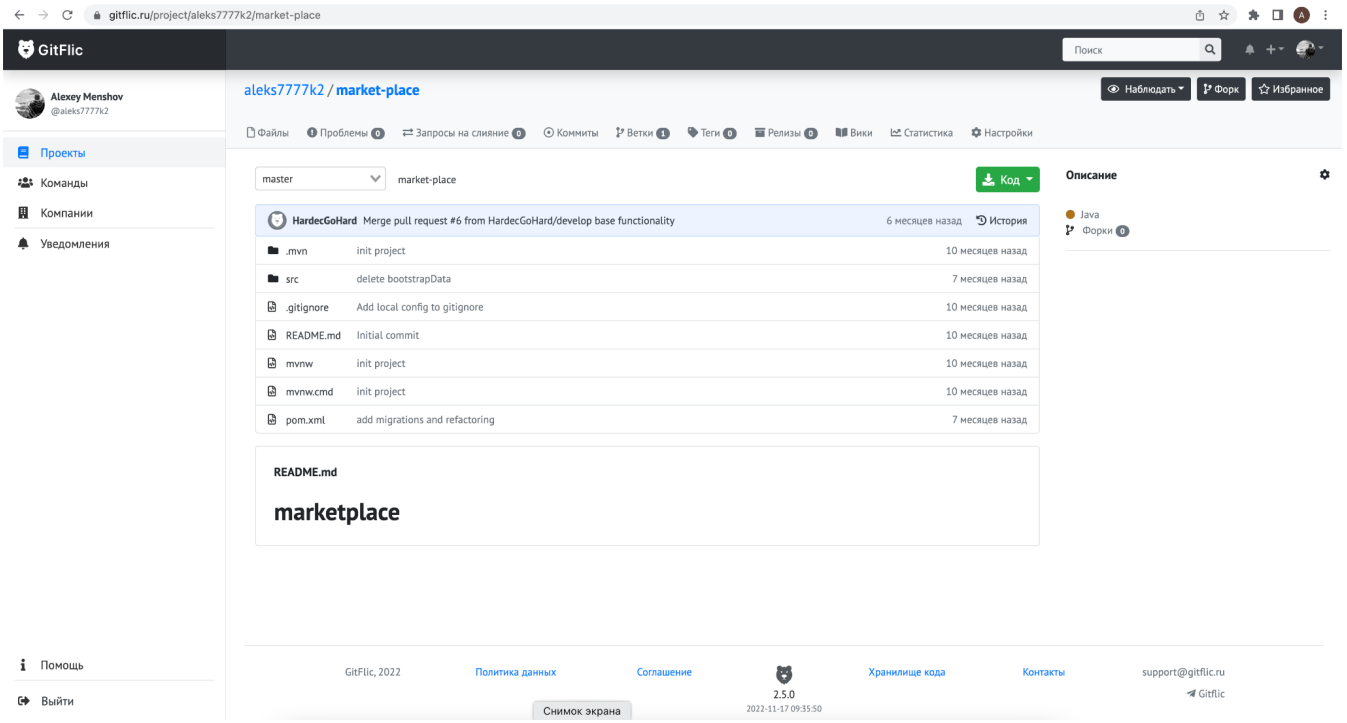

Рисунок 3 – страница проекта

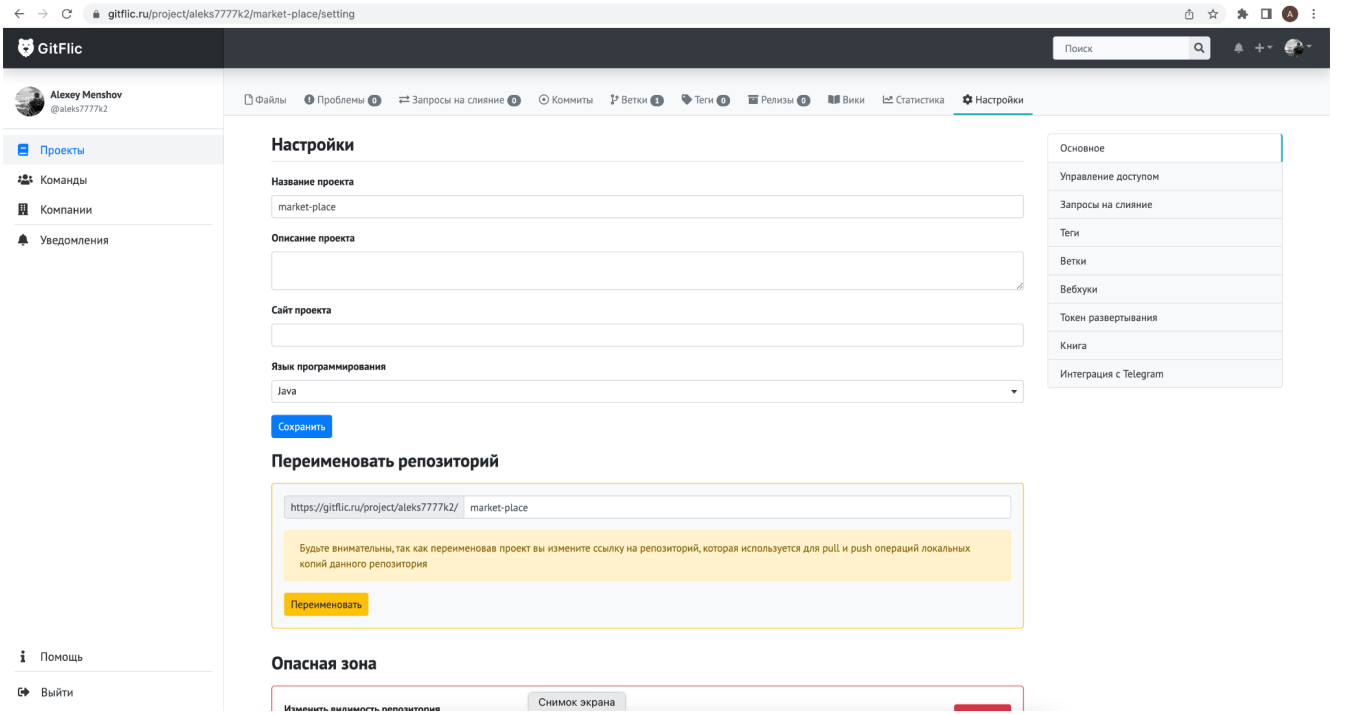

Рисунок 4 – настройка проекта

3. Реализация функционала уведомлениях о событиях в проектах GitFlic в мессенджер Telegram.

В рамках учебной практики основным заданием была реализация системы оповещения о событиях в проекте в мессенджер Telegram.
Для этого необходимо:
Погрузиться в технологический стек проекта GitFlic
Изучить Telegram Bot API
Спроектировать и интегрировать в GitFlic систему оповещения о событиях в мессенджер Telegram.

3.1 Технологический стек

Стек технологий (от англ. stack — «стопка») — это набор технологий, на основе которых разрабатывается сайт или приложение.

3.1.1 Язык программирования Java

Java - строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle. Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины.

3.1.2 Spring Framework

Spring Framework – самый популярный фреймворк для разработки приложений на Java. Spring значительно упрощает разработку, так как реализует в себе IoC(Inversion Of Control) и DI(Dependency Injection).

IoC — это делегирование части наших обязанностей внешнему компоненту.

Dependency Injection – возможность внедрять зависимости.

Spring IoC Контейнер используется для описания любого компонента, который может содержать в себе другие компоненты.

Bean — объект класса, представляющий собой завершенный программный элемент с определенной бизнес-функцией либо внутренней функцией Spring, жизненным циклом которого управляет контейнер бинов.

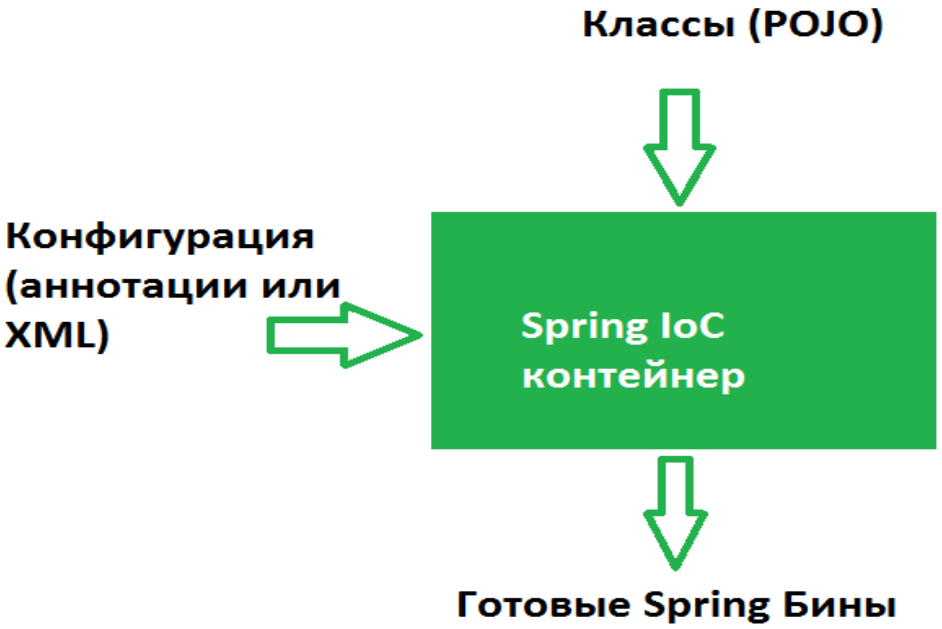

Рисунок 4 – Схема создания Bean’ов Spring фреймворком

3.1.3 Spring Boot

Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

3.1.4 Hibernate

Hibernate позволяет работать с базой данных не напрямую, как это рассмотрено с помощью библиотеки JDBC, а с помощью представления таблиц баз данных в виде классов java.

3.1.5 Spring Data Jpa

Spring Data JPA — это библиотека, которая добавляет дополнительный уровень абстракции поверх ORM реализации JPA. По умолчанию Spring Data JPA использует Hibernate, в качестве ORM провайдера, чтобы выполнять запросы.

3.2 Telegram Bot API.

Telegram Bot API представляет из себя HTTP-интерфейс для работы с ботами в Telegram.

С помощью данного api можно взаимодействовать с телеграм ботами. Telegram предоставляет возможность настроить бота так, чтобы он посылал webhook на определенный URL-адрес при событии/сообщении в чате, участником которого является телеграм бот. Webhook — это «пользовательские обратные вызовы по HTTP». Обычно они запускаются каким-либо событием, например, отправкой кода в репозиторий или комментарием, публикуемым в блоге. Когда происходит это событие, исходный сайт отправляет HTTP-запрос на URL-адрес, указанный для вебхука. Пользователи могут настроить их так, чтобы события на одном сайте вызывали действия на другом.Основной формат — JSON.
Ниже приведен пример исходящей от Telegram API структуры Json:


```
{
 "update_id": 144208320,
 "message": {
  "message_id": 722605,
```

```
    "from": {
      "id": 1365779596,
      "is_bot": false,
      "first_name": "Alexey",
      "last_name": "Menshov",
      "username": "aleks_menshov",
      "language_code": "ru"
    },
    "chat": {
      "id": 1365779596,
      "first_name": "Alexey",
      "last_name": "Menshov",
      "username": "aleks_menshov",
      "type": "private"
    },
    "date": 1668724855,
    "text": "/start",
    "entities": [
      {
        "offset": 0,
        "length": 6,
        "type": "bot_command"
      }
    ]
  }
}
```


Также, Telegram API предоставляет возможность ботам посылать сообщение в чаты. Для этого нужно отправить POST HTTP запрос, в теле которого будут как минимум поле id чата и сообщение которое должен прислать бот.

3.3 Интеграция GitFlic с Telegram Bot API

3.3.1 Постановка задачи

Необходимо разработать систему оповещений о процессе разработки проекта в мессенджер Телеграмм. Пользователь, подключивший интеграцию с телеграмм, должен получать уведомления о создании, редактировании или удалении запросов на слияние, проблем, комментариев и т.д. Также необходимо реализовать фильтрацию событий, конечному пользователю надо предоставить возможность отключать и включать события, которые будут отправлены в телеграм.

3.3.2 Алгоритм интеграции

Перед началом написания кода необходимо решить каким образом будет происходить взаимодействие пользователя с сервисом GitFlic. Для того, чтобы пользователь смог настроить интеграцию с телеграмм был предложен и утверждён следующий алгоритм действий со стороны пользователя:

Рисунок 5 – алгоритм действий со стороны пользователя

После того как пользователь ввел команду боту мы получаем webhook от telegram API в Json формате. Где содержится необходимая информация о сообщении: уникальный идентификатор чата, уникальный идентификатор отправителя, текст сообщения.
Ниже приведен алгоритм работы бота с сообщением

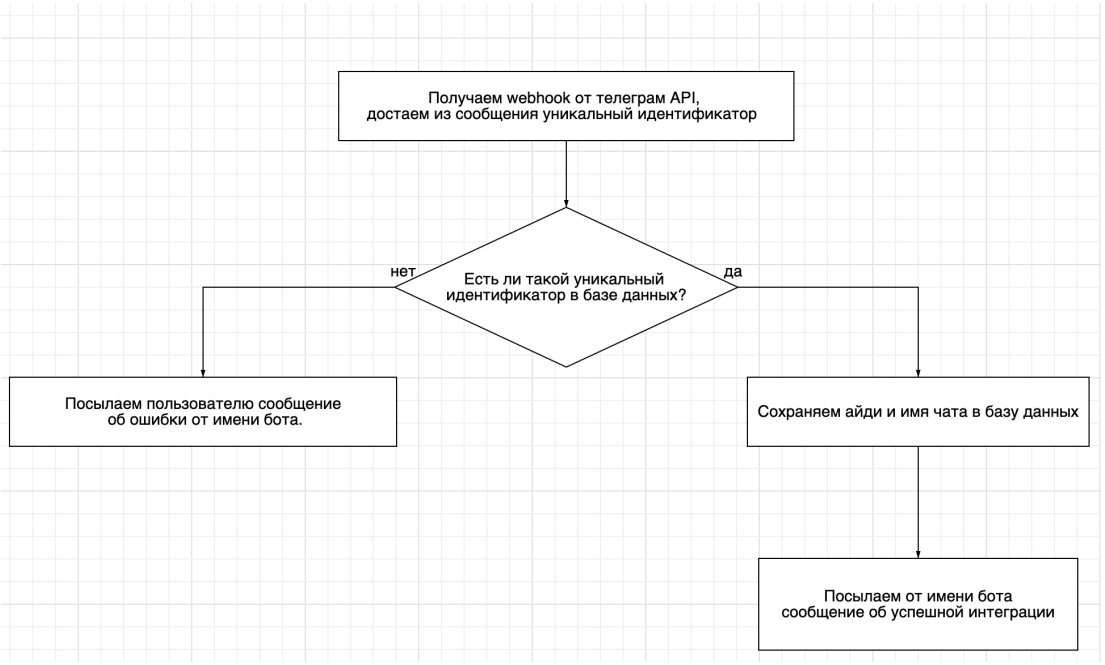

Рисунок 6 – алгоритм валидации токена и сохранения информации о чате в базу

3.3.3 Реализация

Были реализованы следующие страницы:

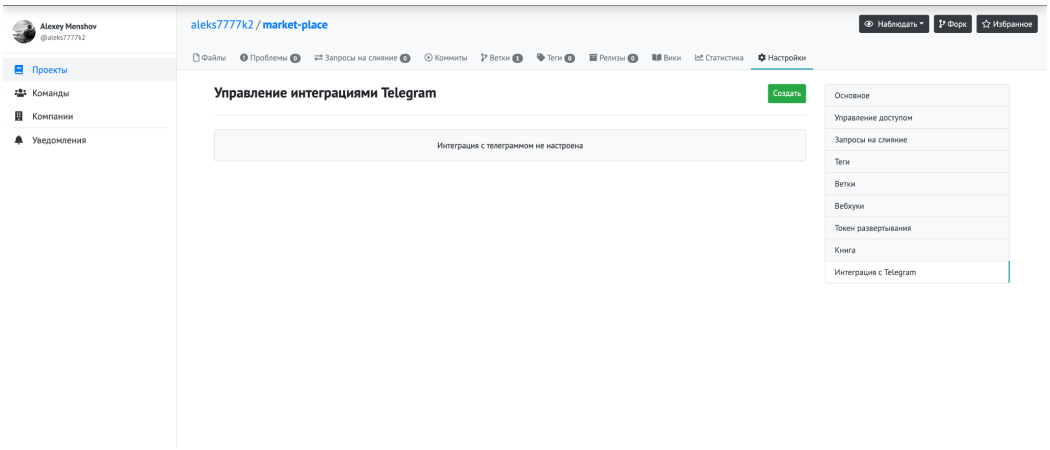

Рисунок 7 – страница создания интеграции с телеграм

При нажатии на кнопку создать пользователь направляется на страницу создания интеграции с Telegram.

Секретный токен - это уникальный идентификатор нужный для интеграции с телеграм ботом.

Список событий для отправки - это фильтрация необходимых событий

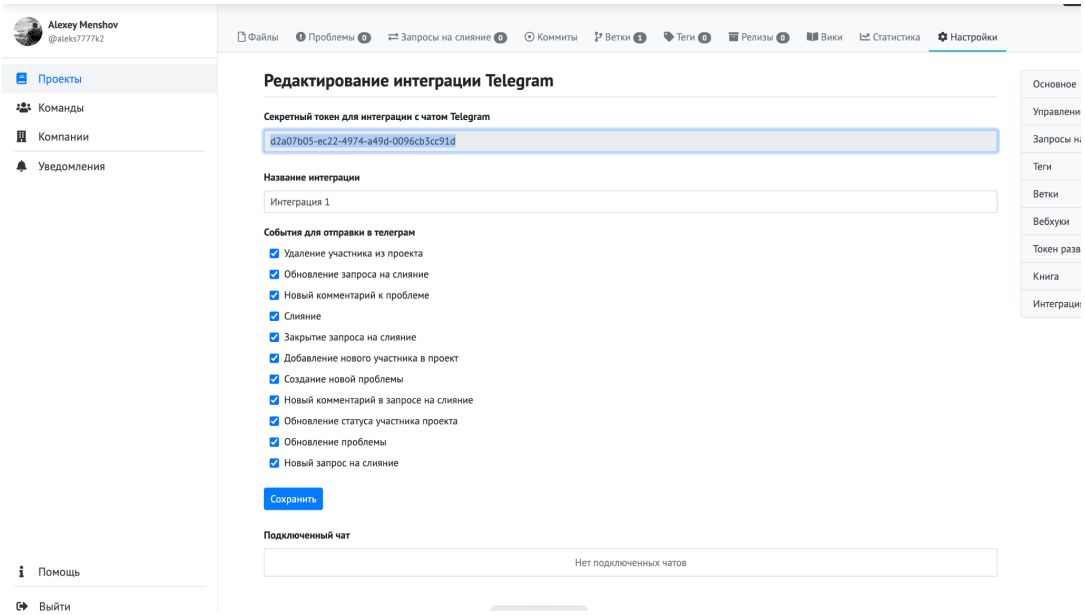

Рисунок 8 – страница конкретной интеграции

После того, как интеграция была успешно создана пользователю необходимо перейти в нужный телеграмм чат с ботом GitFlic и отправить команду /connect <secret token>

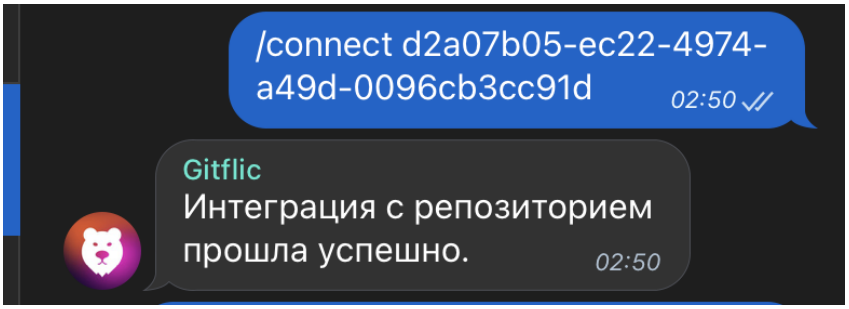

Рисунок 9 – сообщение об успешной интеграции от бота

В случае если токен будет ошибочным то пользователь получит сообщение о том, что такого токена нет:

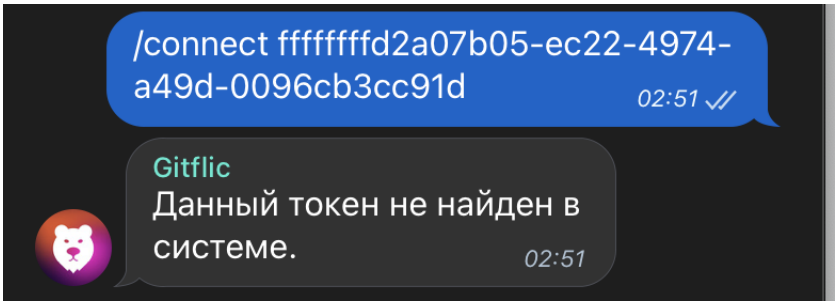

Рисунок 10 – сообщение о неправильном переданном токене

Как только пользователь успешно подключил телеграмм чат на странице с интеграцией появится подключенный чат:

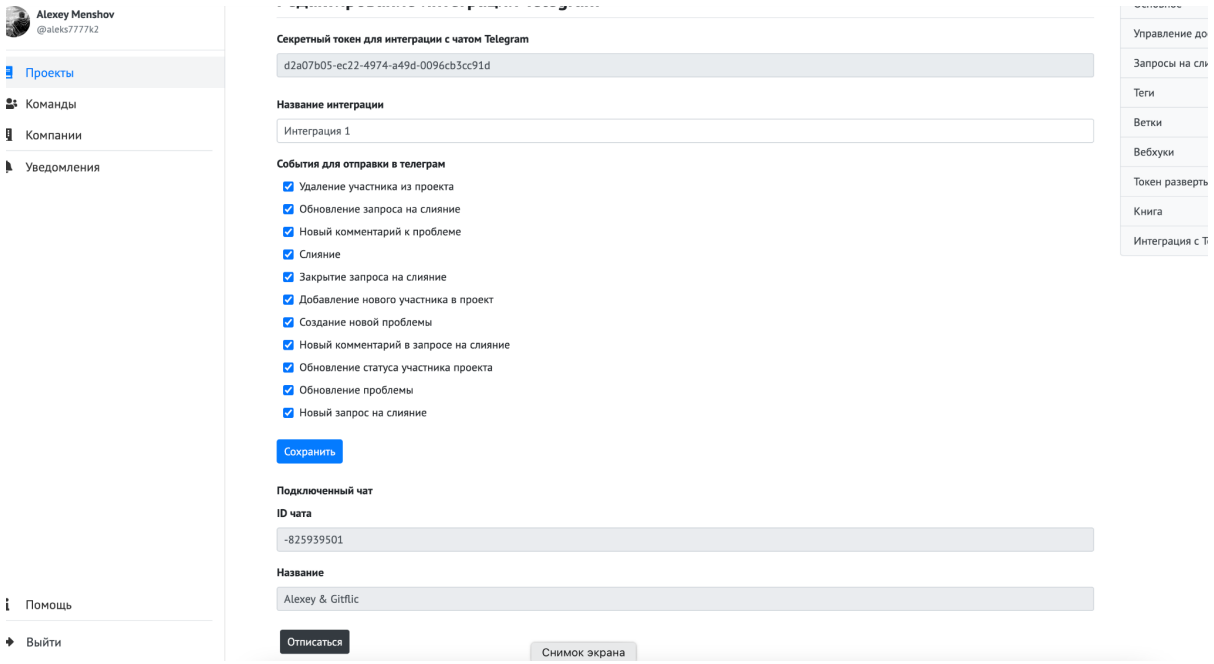

Рисунок 11 – страница конкретной интеграции

3.3.4 Отправка сообщения

При создании, редактировании или удалении чего либо в GitFlic создается событие. Данное событие помещается в очередь событий, на выходе из этой очереди мы можем определить какое именно событие было создано. Для того чтобы отослать правильное сообщение в телеграм чат о событии был реализован паттерн фабричный метод.

Фабричный метод — это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

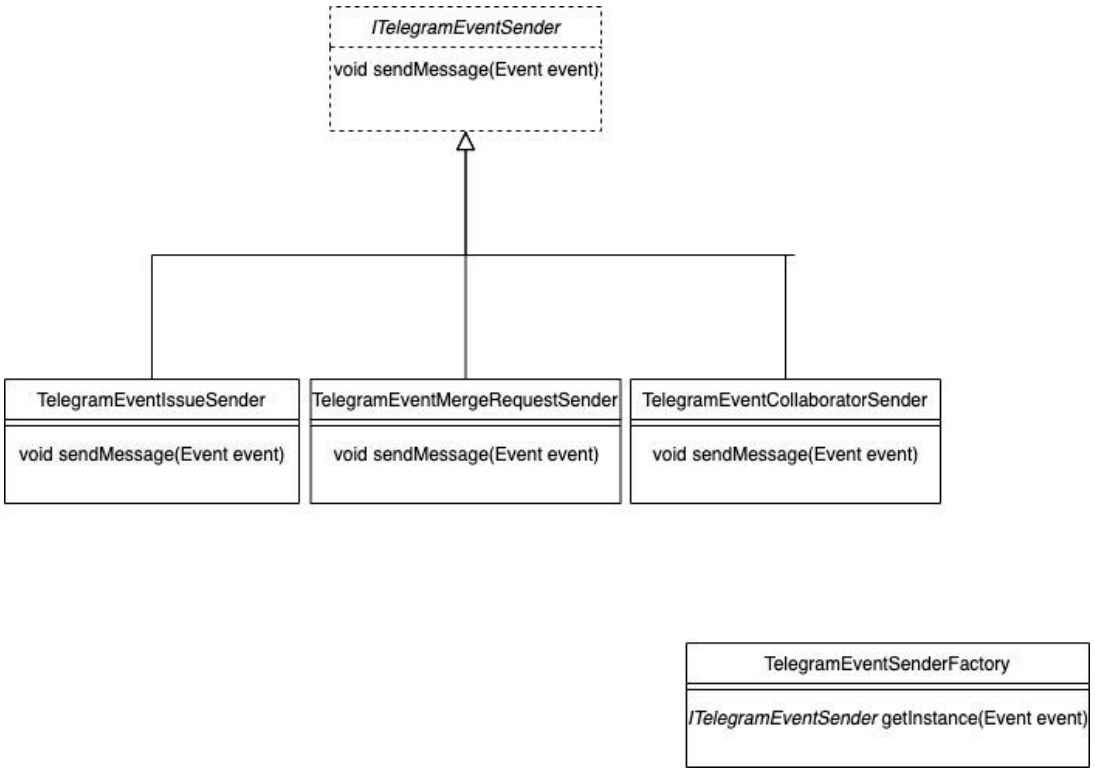

Рисунок 12 – паттерн фабричный метод реализованный для отправки сообщений в телеграм

ITelegramEventSender - интерфейс с методом sendMessage(Event event);

TelegramEventIssueSender - реализация интерфейса ITelegramEventSender отправляет сообщение о изменении или создании проблемы;

TelegramEventMergeRequestSender - реализация интерфейса ITelegramEventSender отправляет сообщение о изменении или создании запроса на слияние;

TelegramEventCollaboratorSender - реализация интерфейса ITelegramEventSender отправляет сообщение о добавлении или удаления участника из проекта;

TelegramEventSenderFactory - класс содержащий в себе метод getInstance(Event event), который возвращает определенную реализацию интерфейса ITelegramEventSender в зависимости оттого какой Event был передан;

Метод send в каждой из реализации интерфейса ITelegramEventSender посылает сообщение в подключенные телеграмм чаты:

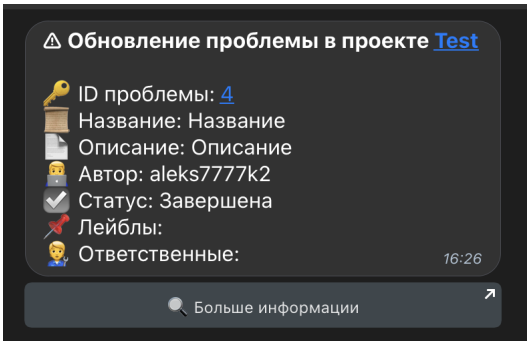

Рисунок 13 – пример сообщения от бота на изменение проблемы в проекте.

Заключение

В процессе прохождения производственной практики были выполнены следующие задачи:

Изучена система работы Git;

Изучена система работы GitFlic;

Реализован функционал уведомлений о событиях в проекте в мессенджер Telegram.

Список литературы

- "Head First Java, Изучаем Java", Кэти Сьерра, Берт Бэйтс
- "Java. Руководство для начинающих", Герберт Шилдт
- Version Control with Subversion
- Документация Telegram Bot API https://core.telegram.org/bots/api
- Spring in Action(5th Edition)