|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и вычислительная техника (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

**ОТЧЕТ ПО УЧЕБНОМУ ПРАКТИКУМУ**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Ментор команды  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Ментор команды **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

*2019 г.*

Оглавление

[Введение 3](#_Toc9361003)

[Аналитический раздел 4](#_Toc9361004)

[Конструкторский раздел 6](#_Toc9361005)

[Технологический раздел 10](#_Toc9361006)

[Личный вклад 14](#_Toc9361007)

[Заключение 16](#_Toc9361008)

[Литература 17](#_Toc9361009)

Введение

Летняя практика направлена на отработку навыков командной разработки, использования системы контроля версий, тестирования, создания документации и программных продуктов, знакомит нас с гибкими методологиями разработки ПО.

Задача нашей команды - создать функционирующего чат-бота на платформе Telegram, помогающего в планировании и распределении как личных, так и командных задач.

Для этого между нами были распределены роли следующим образом:

* Мышонков Максим – руководитель разработки
* Катышев Валентин – разработчик, тестировщик
* Арсений Пронин – разработчик, технический писатель
* Евсигнеев Тимофей –аналитик, тестировщик

Аналитический раздел

На момент создания нашего продукта существуют следующие решения, предоставляющие схожие возможности:

*SkeddyBot* - личные напоминания:

Плюсы:

* Ввод времени вместе с текстом одним сообщением
* Время устанавливается через выбор города
* Циклические уведомления
* Добавление напоминания командой, текстом или кнопкой в интерфейсе
* Есть web-интерфейс со списком
* Мультиязычность

Минусы:

* Нет сортировки задач
* Нельзя добавить несколько задач
* Не работает в групповых чатах
* Не предусмотрено распределение задач для нескольких людей
* Недоступно редактирование текста задачи

*GroceryListBot* - список покупок

Плюсы:

* Доступна сортировка по дате или алфавиту
* Возможно добавление нескольких задач в одно сообщение

Минусы:

* Недоступно редактирование текста задачи
* Нельзя задать срок выполнения
* Только английский язык
* Не работает в групповых чатах

*InstantRemindersBot* - простые напоминания

Плюсы:

* Мультиязычность
* Готовые пресеты таймера
* Может переслать текст, фото, аудио
* Несколько языков на выбор
* Возможно изменение часового пояса

Минусы:

* Нельзя точно задать время
* Нет списка задач
* Нет сортировки задач
* Не работает в групповых чатах

Постановка задачи

Наша цель – создание бота, который будет выгодно отличаться от рассмотренных аналогов. При это будет выполнена задача, разбитая на следующие этапы:

1. Учесть в работе все плюсы и минусы существующих аналогов.
2. Придумать и реализовать удобный и минималистичный интерфейс, используя ограниченный набор инструментов.
3. Проверить программу на ошибки и уязвимости с помощью тестов.
4. Составить документацию и отчет о проделанной работе.

Вывод:

В данный момент не существует продуктов в форм-факторе мессенджер-ботов, позволяющих организовать командный рабочий процесс с распределением задач и настройкой уведомлений. Рассмотренные выше решения не охватывают всю область своего возможного применения. Потому нами решено написать бота, который реализует все тонкости командной работы и исправит найденные нами недостатки других подобных проектов.

Конструкторский раздел

Задачи, поставленные перед каждым участником

Мышонков Максим

1. Организация работы команды
2. Создание документации
3. Настройка сервера
4. Реализация системы автоматического развертывания ПО
5. Проектирование базы данных (БД)

Катышев Валентин

1. Установка требований к системе
2. Создание IDEF0-диаграмм
3. Разработка пользовательского интерфейса
4. Создание пользовательских историй
5. Помощь в разработке

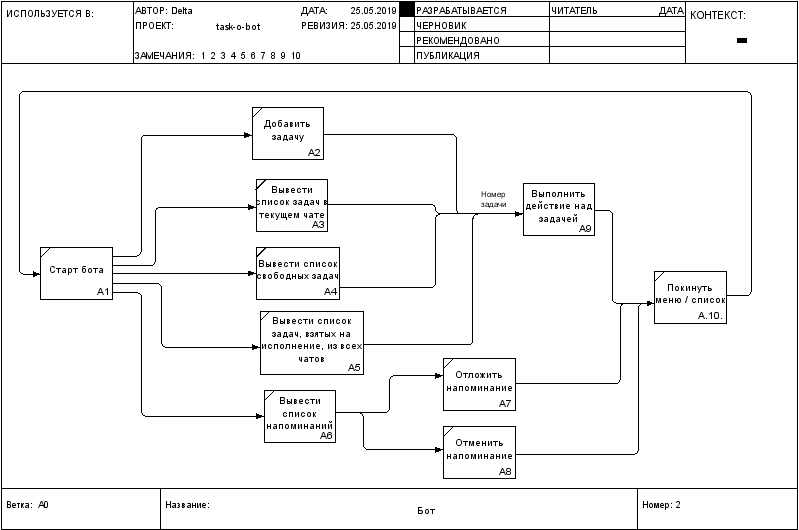
Пронин Арсений

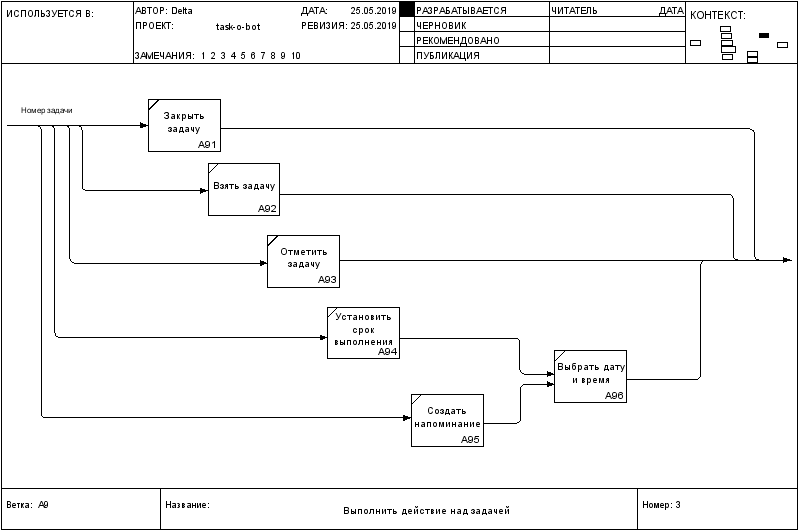
1. Оценка актуальности проекта
2. Выбор технических средств
3. Анализ мнения референтной группы
4. Анализ функциональных особенностей продуктов
5. Тестирование ПО

Евсигнеев Тимофей

1. Поиск аналогов проекта
2. Создание отчетных презентаций
3. Администрирование БД
4. Разработка сценариев тестирования
5. Подготовка первоначального пакета данных для заполнения БД

Декомпозиция задачи.





Бот работает на платформе Telegram. Взаимодействие с ним возможно как в групповых, так и в личных чатах мессенджера. В зависимости от типа чата пользователю доступны те или иные действия.

Бот работает с удаленной базой данных, где для каждой задачи хранятся

1. идентификатор пользователя, создавшего задачу,
2. идентификатор чата, в котором была создана задача,
3. непосредственно сам текст задачи,
4. срок её выполнения,
5. идентификаторы пользователей, назначенных на эту задачу,
6. дата и время созданных напоминаний.

При каждом запросе анализируется идентификатор пользователя на предмет доступа к той или иной настройке задачи, обусловленный либо тем, является ли пользователем создателем задачи, либо заданными или незаданными теми или иными значениями. В связи с этим, пользователю предлагаются только те действия, на которые у него есть доступ.

В групповых чатах доступны следующие действия:

* просмотр списка задач текущего чата (команда /list);
* просмотр списка свободных задач текущего чата (/free);
* создание задачи (/add);
* взятие задачи на исполнение;
* снятие себя с исполнения задачи;
* установка или удаление срока выполнения задачи;
* отметка или снятие отметки приоритетной задачи;
* создание напоминания (после оно отображается лишь в личной беседе);
* закрытие задачи;

Действия без команд производятся через интерфейс с помощью кнопок.

В личных чатах доступны все вышеперечисленные действия, а также следующие:

* просмотр списка напоминаний;
* откладывание напоминаний;
* отмена напоминаний;
* просмотр списка свободных задач из всех чатов пользователя.

Технологический раздел

Технические средства

Язык программирования Python 3

Основной причиной выбора данного языка программирования является его гибкость в использовании и полнота в контексте курса программирования, пройденного в течение первого года обучения. Благодаря этому стало возможным в минимальные сроки начать реализацию проекта практически без затруднений.

Мессенджер Telegram

В качестве платформы для бота был выбран Telegram по причине его наибольшей распространенности относительно других мессенджеров, предоставляющих API для создания чат-ботов (Facebook messenger, Kik и т. д.)

СУБД PostgreSQL

Механизм работы с данной система управления базой данных известен членам нашей команды, при этом PostgreSQL обладает всем необходимым функционалом, обеспечивающим удобство хранения и представления информации. В силу данных причин выбор был остановлен именно на ней.

API Python-Telegram-bot

Данный API является наиболее продвинутым по функционалу среди аналогов (aiogram, telebot), поддерживается официальным разработчиком мессенджера, а также имеет максимально полную и наглядную документацию.

Библиотека для работы с БД Psycorg2

Выбор обусловлен тем фактом, что Psycorg2 является единственной поддерживаемой актуальными версиями PostgreSQL и Python библиотекой для работы с базами данных.

Методология разработки

В качестве методологии разработки ПО была выбрана методология Scrum. Главным ее достоинством является то, что она позволяет в жестко фиксированные, сжатые по времени итерации производить работающее ПО с наиболее приоритетным функционалом. Эта методология идеально подходит для условий учебной практики.

Организация выпуска сборок

Работа со сборками осуществляется при помощи GitLab CI, в которой компоненты программы подвергаются модульному тестированию, после чего, при условии его успешного завершения, происходит развертывание в соответствующей среде (тестовая среда для develop, основной сервер для master).

Реализация программного продукта

Первоначально наша команда реализовала сетевые компоненты – мы подготовили простого эхо-бота, использовав его для настройки взаимодействия серверов Telegram с хостингом. После этого было проведено проектирование базы данных. Данные организованы в виде двух реляционных таблиц – задачи и соответствующие им напоминания.

Имея готовый сервер и БД, мы приступили к реализации главной части – списка задач. Изначально механизм взаимодействия со списком предусматривал отправку команд с соответствующим id задачи и совершаемым действием. Однако в силу неудобства для пользователя интерфейс был заменен на отправляемую ботом клавиатуру, содержащую все доступные действия. В результате стало возможным отправлять информацию индивидуально конкретному пользователю, не засоряя чат лишними кнопками. С целью уменьшения воздействия на чат также применено разделение списка на страницы, отображающиеся по мере необходимости.

Последний этап разработки – добавление фонового процесса, производящего мониторинг списка уведомлений и отправляющего сообщения пользователям по наступлении соответствующего времени. При отправке уведомления оно не полностью стирается из базы данных, а отмечается выполненным. Такой подход позволил добавить функционал отправки такого же уведомления через некое время, если пользователь считает повторение нужным.

Реализация тестирования

Тестирование проводится при помощи модульных тестов, покрывающих сценарии взаимодействия куратора бота с базой данных, а также процесс добавления и удаления задач. Для этого используется библиотека Python Unittest, обладающая функционалом имитации методов, работа которых завязана на обращении к Telegram. Каждый проверочный модуль (модули проверки назначения задачи пользователю, изменения срока исполнения и т. д.) включает в себя 5 – 8 сценариев, содержащих в свою очередь до 10 проверяемых утверждений, тестирующих корректность работы подпрограммы.

Части программы, недоступные модульному тестированию (те, что напрямую взаимодействуют с пользователем или серверами Telegram) были протестированы вручную посредством комбинации сценариев взаимодействия как в режиме одиночного чата, так и в случае работы в беседе с несколькими пользователями одновременно. Особое внимание уделялось сохранению работоспособности при прерывании соединения с клиентом ввиду наличия проблем со стабильной работой мессенджера у многих людей.

Развертывание разработанного продукта

Для локального запуска бота необходимо настроить значения локальных переменных BOT\_TOKEN (токен бота в telegram) и DATABASE\_URL (схема доступа к базе данных), установить необходимые зависимости для python 3.7 (файл requirements.txt), после чего запустить main.py.

Для функционирования бота необходим доступ к telegram.org, поэтому может понадобиться дополнительная настройка прокси или VPN.

Технологические выводы

Процесс разработки подтвердил обоснованность выбора технических средств, а методология Scrum позволила в сжатые сроки создать продукт, отвечающий нашим ожиданиям. Гибкость языка и прикладного инструментария вкупе с последовательным подходом к разработке – это то, что определило успех каждого спринта нашей команды. Полученный продукт прост в развертывании и разбит на модули без перекрестных зависимостей, что обеспечивает возможность поддержки и дальнейшего развития.

Личный вклад

Одной из основных моих задач стала реализация прикладного инструмента, обеспечивающего доступ к информации о задаче в базе данных. В качестве основного инструмента для этого я использовал библиотеку Psycorg2, направляющую SQL запросы в базу данных PostgreSQL. Для упрощения решения я декомпозировал исходную задачу на 2 этапа. Сначала информация о модификации поступает в подпрограмму модуля db\_connector, выделенную для конкретного действия (назначить исполнителя, срок и т. д.). Функция формирует SQL запрос с соответствующими параметрами, после чего вызывает подпрограмму второго этапа – db\_connector.\_connect. Этот универсальный блок исполняет запрос в рамках инициированного сеанса связи с базой данных, после чего фиксирует изменения и возвращает результат операции.

Затруднение при выполнении данной задачи возникло в связи с тем, что первоначальная реализация инициировала новое подключение к базе данных при выполнении каждой операции, что создавало дополнительную нагрузку и увеличивало время обработки. Решением данной проблемы стал объектно-ориентированный подход к программированию. Для выполнения операций, связанных с базой данных, мною был создан класс, включающий в себя все необходимые методы, а также объект типа Psycorg.connection, позволяющий открывать и закрывать соединение по мере необходимости. Таким образом, при взаимодействии с базой данных подключение создается один раз, закрываясь после выполнения всех необходимых в данных момент манипуляций.

Вторая из выполненных мной задач – это создание механизма отправки уведомлений пользователям. К любой задаче из доступных списков через меню взаимодействия возможно добавить одно или несколько напоминаний, после чего в выбранное время бот отправит сообщение с соответствующей информацией. Используя подготовленную командой таблицу базы данных, содержащую информацию о уведомлениях, я разработал подпрограмму, обрабатывающую все записи с истекшим временем. После сбора информации о соответствующих задачах происходит рассылка сообщений пользователям. К каждому сообщению прикрепляются две кнопки, позволяющие перенести уведомление на другое время или окончательно закрыть его.

Проблемой в данной схеме стала возможность закрытия задач. При этом должны отменяться все уведомления о ней. Так как при закрытии задачи запись о ней не удаляется сразу же из базы (на случай возможной отмены этого действия), реализовать отмену уведомлений через каскадное удаление в PostgreSQL не представлялось возможным. Для исправления ситуации пришлось прибегнуть к созданию функции внутри SQL запроса закрытия задачи. Это позволило получить ее id, после чего по данному значению производится отмена всех связанных напоминаний.

В дополнение к вышеописанному функционалу я реализовал систему менеджмента уведомлений, доступную в личной беседе. При вызове команде /rem пользователь получает список всех напоминаний, которые он должен получить в будущем. Для упрощения взаимодействия с пользователем я выделил каждую запись в отдельное сообщение, к которому привязаны две кнопки, позволяющие отменить или перенести уведомление.

С целью уменьшить засорение диалога при отмене напоминания происходит удаление более не нужных сообщений, выполняемое посредством анализа возвращаемой информации, которая содержит id соответствующего сообщения.

Заключение

В течение работы нам пришлось изменять некоторые первоначальные планы и находить альтернативы для некоторых функций.

К примеру, первым возникшим затруднением стала невозможность обращения и получения id пользователя по его имени, что является одним из недостатков всех современных ботов Telegram и, возможно, других мессенджеров.

Проблема неактивности web-worker’a после получаса использования была решена изменением на python-worker, который сервер принимает за обычный скрипт и соединение не прерывается. К сожалению, из-за этого пришлось убрать технологию webhooks.

Декомпозиция задач и чёткая формулировка позволила добиться удобства разработки и создания качественного кода, разделенного на модули, и упрощения разработки и тестирования.

Полное интеграционное тестирование бота было затруднено из-за особенностей проекта, т. е. использования стороннего программного обеспечения и отсутствия доступа к программному коду приложения-мессенджера и его серверам.

Так как бот первоначально разрабатывался для групповых чатов, он довольно успешно справляется с назначением задач от нескольких пользователей и стабильно работает при изменении этих задач. Также он работает в личных чатах с пользователями. Одно из преимуществ подобного проекта – доступность для пользователя, что помогает тестировать бота в максимально приближенных к жизни условиях на каждом этапе разработки.

За время практики каждый из нас получил опыт командной разработки по технологии Scrum и получил новые знания о создании бота для мобильной платформы и структуре подобных проектов, познакомился с новыми библиотеками, которые позволили значительно упростить и ускорить процесс разработки, улучшил стиль написания кода.

Литература

1. Jones Brian K. Python Cookbook / Brian K. Jones. - O'Reilly Media, 2013 – 706 с.
2. Lutz Mark.Python Pocket Reference: Python In Your Pocket (Pocket Reference (O'Reilly)). / Mark Lutz. – 5-е изд. –O'Reilly Media, 2014 – 266 с.
3. Официальная документация Python 3.7 [Электронный ресурс]. – URL: <https://docs.python.org/3.7/> (дата обращения: 20.05.2019).
4. Официальная документация Python Telebram Bot 12.0.0b1 [Электронный ресурс]. – URL: <https://python-telegram-bot.readthedocs.io/en/latest/> (дата обращения: 20.05.2019).
5. Официальная документация Psycopg 2.8 [Электронный ресурс]. – URL: <http://initd.org/psycopg/docs/> (дата обращения: 20.05.2019).