Geekbrains

**Создание бота для Telegram на Java с использованием Telegram Bot API.**

IT-специалист:

Гасанов Н.Н.

Москва

2024

Оглавление

[Введение 3](#_Toc155737842)

[Глава 1 Теоретические основы 5](#_Toc155737843)

[1.1 Концепция и виды ботов 5](#_Toc155737844)

[1.2 Особенности и преимущества Telegram, как платформы для ботов 10](#_Toc155737845)

[1.3 Описание и возможности Telegram Bot API 14](#_Toc155737846)

[Глава 2 Выбор технологий и среды разработки 19](#_Toc155737847)

[2.1 Обоснование выбора Java как языка программирования 19](#_Toc155737848)

[2.2 Сравнение и выбор Spring boot как фреймворка для разработки веб-приложений 22](#_Toc155737849)

[2.3 Выбор и настройка PostgreSQL как базы данных для хранения информации о пользователях и бронированиях 25](#_Toc155737850)

[Глава 3 Разработка телеграмм-бота для бронирования 28](#_Toc155737851)

[3.1 Описание функциональности и интерфейса бота 28](#_Toc155737852)

[3.2 Реализация основных методов и классов бота 34](#_Toc155737853)

[3.3 Тестирование и отладка бота 36](#_Toc155737854)

[3.4 Документирование кода и работы бота 38](#_Toc155737855)

[Заключение 41](#_Toc155737856)

[Список использованных источников 42](#_Toc155737857)

[Приложение 43](#_Toc155737858)

# Введение

В современном мире информационных технологий и цифровой коммуникации все большую популярность и востребованность приобретают боты — программы, которые автоматически выполняют определенные задачи, общаются с пользователями и предоставляют им различные сервисы. Боты могут работать в разных средах и платформах, таких как веб-сайты, мессенджеры, социальные сети, игры и т.д. Одной из самых популярных и удобных платформ для создания и использования ботов является Telegram — кроссплатформенный мессенджер, который обладает множеством преимуществ, таких как высокая скорость, безопасность, простота, функциональность и гибкость. Telegram предоставляет разработчикам специальный интерфейс — Telegram Bot API, который позволяет создавать ботов для Telegram с помощью различных языков программирования и фреймворков.

Целью данной дипломной работы является разработка телеграмм-бота для бронирования на Java с использованием Telegram Bot API. Данный бот предназначен для того, чтобы помогать пользователям легко и быстро бронировать различные ресурсы и услуги в организации, такие как фитнес-зал, оборудование, машины и т.д. Бот должен иметь удобный и интуитивный интерфейс, обрабатывать запросы и команды пользователей, предоставлять им информацию и опции для выбора, подтверждать и отменять бронирования, отправлять уведомления о записи и т.д. Бот должен также хранить информацию о пользователях и бронированиях в базе данных и обеспечивать ее безопасность и целостность.

Для реализации данной цели в работе будут использованы следующие методы исследования: анализ теоретических основ и практических примеров создания ботов для Telegram, выбор и обоснование технологий и среды разработки, проектирование и реализация бота с использованием Java и Telegram Bot API, тестирование и отладка бота, документирование кода и работы бота, анализ результатов и выводов по работе.

Структура работы состоит из следующих частей: введение, три главы, заключение, список использованных источников и приложение. В первой главе будут рассмотрены теоретические основы и виды ботов, особенности и преимущества Telegram как платформы для ботов, описание и возможности Telegram Bot API, анализ существующих ботов для бронирования. Во второй главе будут описаны выбор и настройка технологий и среды разработки, таких как Java, Spring boot, PostgreSQL и Heroku, а также разработка архитектуры и дизайна бота. В третьей главе будут представлены реализация, тестирование и документирование бота, а также анализ его функциональности и интерфейса. В заключении будут подведены результаты и выводы по работе, а также указаны направления для дальнейшего развития и улучшения бота.

Важно подчеркнуть, что актуальность и цель исследования, которое определяет объект и предмет исследования, формулирует основные задачи и методы исследования, а также описывает структуру работы, демонстрируя при этом знание темы исследования, а также свою способность логично и последовательно излагать свои мысли и аргументы.

# Глава 1 Теоретические основы

# Концепция и виды ботов

Бот (от английского "robot") — это программное обеспечение, способное автоматически выполнять определенные задачи или взаимодействовать с пользователями через различные каналы коммуникации, такие как текстовые сообщения или голосовые команды. Функционирование бота обычно основана на алгоритмах и искусственном интеллекте, позволяющих ему анализировать и понимать ввод пользователя, обрабатывать запросы и предоставлять соответствующую информацию или выполнять действия. Боты могут использовать предопределенные шаблоны ответов или обучаться на основе машинного обучения и нейронных сетей для улучшения своей производительности и адаптации к потребностям пользователей.

Основные задачи, которые могут быть решены с помощью ботов:

* Коммуникация и поддержка клиентов. Боты могут предоставлять помощь и отвечать на вопросы пользователей, осуществлять бронирование, предоставлять информацию о товарах или услугах, обрабатывать жалобы и так далее.
* Автоматизация и оптимизация бизнес-процессов. Боты могут выполнять рутинные задачи, такие как заполнение форм, обработка заказов, отправка уведомлений и автоматическое управление системами.
* Развлечение и информационные сервисы. Боты могут предоставлять пользователю развлекательные услуги, игры, мемы, новости, погоду, финансовую информацию и многое другое.

Какой есть ряд преимущества у такого инструмента, как бот?

* Доступность и удобство. Боты могут быть доступны 24/7 и на различных платформах, что обеспечивает удобство использования для пользователей.
* Эффективность и автоматизация. Боты способны автоматизировать выполнение задач, что позволяет сэкономить время и ресурсы.
* Масштабируемость. Боты могут обслуживать одновременно большое количество пользователей без необходимости дополнительных ресурсов.
* Повышение уровня обслуживания клиентов. Боты могут предоставлять быстрые ответы и решения на вопросы пользователей, обеспечивая более высокий уровень обслуживания.
* Персонализация и адаптация. Боты могут использовать данные о пользователях для предоставления персонализированного опыта и рекомендаций.

Безусловно есть и недостатки, о которых обязательно стоит упомянуть и над решениями которых прямо сейчас бьются программисты совместно с инженерами всего мира, а именно:

* Ограниченность в понимании. Боты могут сталкиваться с трудностями в понимании и обработке сложных запросов или нестандартных ситуаций.
* Недостаток эмоциональной составляющей. Боты могут быть менее эмоционально отзывчивыми и неспособными к эмпатии, что может негативно сказываться на некоторых видах взаимодействия.
* Ограниченность в способностях. Боты могут быть ограничены в своих функциях и способностях, особенно если они не обладают достаточным уровнем умения или обучения.
* Зависимость от технологических средств. Боты могут требовать устойчивого интернет-соединения и обновлений программного обеспечения для надлежащей работы.

В целом, боты представляют собой мощный инструмент для автоматизации и облегчения различных задач. Они выполняют разнообразные функции и могут быть использованы в различных областях, включая бизнес, развлечения и образование и т.д. Однако, для достижения наилучших результатов, необходимо тщательно разработать и настроить бота с учетом конкретных потребностей и целей.

Работа ботов основана на заранее определенных правилах и логике, а также на алгоритмах искусственного интеллекта. Они могут использоваться для автоматизации рутинных задач, предоставления информации, обработки заказов, поддержки клиентов, развлечения пользователей и т. д.

Так как бот, достаточно гибкий, развивающийся и неподдающийся унификации инструмент, то достаточно сложно говорить о классификации ботов, но давайте поговорим о наиболее распространенных:

1. По способу взаимодействия.

* Текстовые боты - пользователь взаимодействует через текстовый интерфейс, например, чат-окна или мессенджеры.
* Голосовые боты - пользователь взаимодействует голосом, используя технологии распознавания речи и синтеза речи.
* Графические боты - взаимодействуют с пользователем через графический интерфейс, такой как кнопки, изображения и элементы управления.

1. По целям и функциям:

* Информационные боты – предназначены для предоставления информации пользователю, например, справочные боты, новостные боты и т. д.
* Сервисные боты - предназначены для выполнения определенных сервисных функций, например, боты для заказов, бронирования, покупок и т. д.
* Развлекательные боты - созданны для развлечения пользователей, например, игровые боты или боты для создания мемов и шуток.
* Социальные боты - помогают пользователям взаимодействовать с социальными сетями, например, боты для автоматической публикации контента или отслеживания активности пользователей.

1. По уровню сложности и искусственному интеллекту:

* Простые (правила-основанные) боты - используют заранее заданные правила и шаблоны для обработки запросов пользователей и предоставления ответов.
* Боты машинного обучения (ML) - используют алгоритмы машинного обучения для анализа данных и повышения своей производительности со временем. Они могут использовать методы обработки естественного языка (Natural Language Processing, NLP) для понимания и генерации текста.
* Боты глубокого обучения (Deep Learning) - основаны на глубоких нейронных сетях и используют сложные алгоритмы глубокого обучения для обработки данных и принятия решений. Они могут достичь высокой степени автоматизации и понимания естественного языка.

1. По применению:

* Бизнес-боты - используеются в коммерческих целях, например, для предоставления поддержки клиентам, обработки заказов или автоматизации рабочих процессов.
* Медицинские боты - предоставляют медицинскую информацию и помощь, например, для диагностики симптомов или предоставления советов по здоровью.
* Образовательные боты - используются в образовательных

1. По типу автоматизации:

* Полуавтоматические боты - работают в паре с человеком и выполняют определенные задачи с его помощью. Например, чат-боты, которые направляют запросы пользователей к живым операторам для более сложных или специализированных вопросов.
* Полностью автоматические боты - функционируют без участия человека и могут автоматически обрабатывать запросы и предоставлять ответы или выполнить определенные действия.

Классификация ботов может варьироваться в зависимости от конкретной ситуации и контекста их использования. Новые типы ботов и классификации могут появляться с развитием технологий и потребностей рынка. Также, мы не можем не поговорить о тех видах ботов, которые сейчас занимают лидирующую позицию в настоящее время. Давайте остановимся на видах ботов более подробно:

* Чат-боты (или текстовые боты). Это один из наиболее распространенных видов ботов. Они взаимодействуют с пользователями через текстовые сообщения и обычно используются для предоставления информации, ответов на вопросы, выполнения задач и т. д. Примеры чат-ботов включают боты для поддержки клиентов, боты для бронирования билетов или отелей, а также персональные помощники, такие как Siri и Google Assistant.
* Голосовые боты. Данные боты взаимодействуют с пользователями через голосовые команды и ответы. Они обычно используются в устройствах, таких как умные домашние ассистенты (например, «Яндекс.Станция» со встроенным голосовым помощником «Алиса) или встроены в мобильные приложения. Голосовые боты могут выполнять различные задачи, включая поиск информации, управление устройствами, чтение новостей и многое другое.
* Роботы-ассистенты. Они обычно представляются в виде роботов, которые могут перемещаться и выполнять физические задачи. Они могут быть использованы в сферах, таких как розничная торговля, гостиничный бизнес или медицина. Роботы-ассистенты могут помогать клиентам, обслуживать заказы, предоставлять информацию и даже выполнять некоторые сложные задачи, требующие физического взаимодействия.
* Торговые боты. Как правило — это автоматизированные программы, которые выполняют торговые операции на финансовых рынках. Они используются для выполнения различных стратегий торговли, таких как арбитраж, маркет-мейкинг или анализ рынка. Торговые боты обычно основаны на алгоритмах и могут работать самостоятельно без участия человека.
* Социальные боты. Они очень активны в социальных медиа и выполняют различные задачи, например такие как автоматизированное публикование контента, ответ на комментарии и сообщения, анализ данных и т. д.
* Игровые боты. Созданы для игровых целей и могут быть как автономными, так и взаимодействовать с другими игроками. Они могут управлять NPC персонажами в игровом мире, предлагать задания или помогать игрокам в прохождении игры.

Приведенные выше примеры являются лишь некоторыми из многочисленных видов ботов, которые существуют в настоящее время. Технологии ботов продолжают развиваться, и с каждым днем появляются новые возможности и применения для ботов.

Итак, какие выводы мы можем сделать? Прежде всего при разработке любого вида бота вам необходимо определить цель, задачи и функциональность, ведь исключительно четко и точно поставленная цель сможет дать толчком правильному исходу, но также не стоит забывать о сложности поставленной задачи, обработке источников, типе автоматизации и возможности обновления вашего бота.

# 1.2 Особенности и преимущества Telegram, как платформы для ботов

Telegram (от др.-греч. τῆλε «далеко» + др.-греч. γράμμα «запись») — это кроссплатформенная система мгновенного обмена сообщениями (мессенджер), предоставляющая функции обмена текстовыми, голосовыми и видеосообщениями, а также стикерами, фотографиями и файлами различных форматов. Он также поддерживает аудио и видеозвонки, трансляции в каналах и группах, конференции, многопользовательские группы и каналы [1]. Приложение Telegram имеет клиентские версии для Android, iOS, Windows, macOS и GNU/Linux.

Данный мессенджер предлагает расширенную функциональность путем использования ботов. Боты позволяют значительно расширить возможности приложения, добавляя новые функции и сервисы. Они могут автоматизировать различные задачи, предоставлять информацию, играть и интерактивно взаимодействовать с пользователями.

Telegram является одним из самых загружаемых приложений, и в июне 2022 года оно вошло в пятёрку самых популярных приложений. Количество постоянных пользователей Telegram превысило 700 миллионов. По словам основателя сервиса Павла Дурова, к началу 2023 года Telegram стал вторым по популярности мессенджером в мире, уступая только WhatsApp. Стоит отметить, что Telegram — это платформа с широким спектром функций обмена сообщениями, поддерживающая ботов и пользующаяся большой популярностью среди пользователей по всему миру.

Большая аудитория мессенджера также означает, что боты могут быть применены в различных сферах деятельности, таких как коммерция, образование, развлечения, здравоохранение и другие. Разработчики могут создавать ботов, которые предлагают услуги, отвечают на вопросы пользователей, предоставляют информацию, развлекают и автоматизируют различные задачи. Благодаря широкой аудитории Telegram, боты имеют потенциал привлечь большое количество пользователей и стать незаменимыми инструментами для взаимодействия с ними.

Таким образом, огромная популярность Telegram и его широкий географический охват делают его привлекательной платформой для разработки ботов. Разработчики могут использовать эту платформу для достижения миллионов пользователей и предоставления им уникального и интересного опыта взаимодействия с ботами.

Мессенджер предлагает мощный и гибкий набор функций и возможностей для разработки и использования ботов. Боты в Telegram могут взаимодействовать с пользователями, предоставлять информацию, выполнять задачи и автоматизировать различные процессы.

Одной из основных функций ботов в Telegram является обмен текстовыми сообщениями. Боты могут принимать и отправлять текстовые сообщения пользователям, обрабатывать команды, отвечать на вопросы и предоставлять информацию.

Кроме текстовых сообщений, боты могут работать с различными медиа-файлами. Они могут отправлять и принимать изображения, аудио- и видеофайлы, документы различных форматов. Таким образом, боты могут обмениваться мультимедийным контентом с пользователями, предоставлять им файлы и другие материалы.

Telegram также предоставляет возможность создания интерактивных элементов в ботах. Боты могут использовать кнопки и клавиатуры, чтобы предоставить пользователям выбор, взаимодействовать с ними и упростить использование бота. Кнопки могут быть настроены для выполнения определенных действий, вызова команд или перехода на определенные страницы.

Кроме работы с отдельными пользователями, боты в Telegram могут также взаимодействовать с группами и каналами. Они могут быть добавлены в группы и каналы, чтобы предоставлять дополнительные функции и сервисы. Боты могут отправлять уведомления, проводить опросы, отвечать на вопросы участников и выполнять другие задачи, связанные с групповым взаимодействием.

Telegram также предоставляет возможность интеграции с внешними сервисами. Боты могут использовать API других платформ и сервисов для получения данных, отправки уведомлений, выполнения операций и других функций. Это позволяет разработчикам интегрировать ботов с существующей инфраструктурой и расширить их возможности.

В целом, Telegram обеспечивает разработчикам ботов широкий набор функций и возможностей и именно это делает его привлекательной платформой для создания мощных и интерактивных ботов.

Безопасность является одним из важнейших аспектов в мессенджере Telegram, и сервис предпринимает множество мер для обеспечения конфиденциальности переписки и защиты данных пользователей.

Мессенджер использует протокол шифрования MTProto, который разработан с учетом высоких стандартов безопасности. Этот протокол обеспечивает конфиденциальность сообщений, а также защиту от несанкционированного доступа и перехвата данных. Все сообщения в Telegram шифруются на устройстве отправителя и дешифруются только на устройстве получателя. Это означает, что даже если данные были перехвачены в процессе передачи, они остаются зашифрованными и недоступными для посторонних лиц.

Кроме того, Telegram предлагает функцию "секретных чатов", которая обеспечивает еще большую степень конфиденциальности. В секретных чатах используется дополнительный уровень шифрования, и сообщения могут быть настроены на самоуничтожение через определенное время после прочтения. Это позволяет пользователям обмениваться сообщениями с высокой степенью секретности и уверенности в их удалении после прочтения.

Telegram также активно борется с попытками несанкционированного доступа к аккаунтам пользователей. Он предлагает функцию двухфакторной аутентификации, которая позволяет пользователям установить дополнительный пароль для защиты своего аккаунта. Это добавляет дополнительный уровень безопасности и предотвращает несанкционированный доступ даже в случае утечки основного пароля.

Важно отметить, что Telegram также предлагает функцию "конфиденциальных чатов", которая позволяет пользователям удалять сообщения не только с их собственных устройств, но и с устройств собеседников. Это может быть полезно при необходимости удаления конфиденциальной информации или после окончания переписки.

В целом, Telegram придает большое значение безопасности и приватности пользователей и обеспечивает высокий уровень защиты данных и конфиденциальности в мессенджере. Это делает его одним из самых безопасных мессенджеров на рынке.

Telegram активно развивается и обновляется, предлагая новые функции и улучшения как для обычных пользователей, так и для разработчиков ботов. Это делает платформу более мощной и гибкой в использовании.

Данный мессенджер поддерживает активное сообщество разработчиков, которые создают и поддерживают ботов на платформе. Существует множество ресурсов, форумов, групп и каналов, где разработчики могут обмениваться знаниями, делиться опытом и получать поддержку от других членов сообщества. Это позволяет разработчикам обмениваться идеями, решать проблемы и создавать лучшие боты.

Telegram также предоставляет разработчикам полезные инструменты и ресурсы для создания и развертывания ботов. Они предлагают Telegram Bot API, который позволяет разработчикам взаимодействовать с платформой и создавать ботов с помощью различных языков программирования.

Также разработчики регулярно выпускают обновления, в которых предлагаются новые функции и улучшения. Это может включать расширенные возможности для ботов, улучшенные инструменты для разработчиков, улучшенную производительность и безопасность. Обновления позволяют разработчикам использовать новые функции и инструменты для создания более интересных и полезных ботов.

В целом, благодаря активному сообществу разработчиков, поддержке со стороны Telegram и регулярным обновлениям, платформа продолжает развиваться и совершенствоваться. Это создает благоприятную среду для разработки и поддержки ботов на платформе Telegram.

# 1.3 Описание и возможности Telegram Bot API

В настоящее время, несмотря на различные споры, Telegram является тем мессенджером, который задает различные инновационные тренды в экосистеме обмена мгновенных сообщений, в связи с чем он предоставил нам набор определенных правил и протоколов, которые позволяют различным программным приложениям взаимодействовать друг с другом, так называемые Application Programming Interface или коротко API. Нужно сказать, что API определяет способы, как одно программное приложение может запросить данные или выполнить определенные операции у другого приложения или сервиса и исходя из этого мы можем сказать, что Telegram Bot API — это набор интерфейсов и методов, предоставляемых Telegram, для разработки и взаимодействия с ботами в мессенджере Telegram. Telegram Bot API позволяет разработчикам создавать ботов, которые могут выполнять различные задачи и функции, взаимодействовать с пользователями и получать доступ к различным функциональным возможностям Telegram [2].

С помощью Telegram Bot API разработчики могут отправлять и получать сообщения, изображения, аудио, видео и другие медиафайлы, создавать клавиатуры и кнопки для взаимодействия с пользователями, получать информацию о пользователях и группах, управлять подписками на уведомления и многое другое.

API предоставляет различные методы, которые можно использовать для отправки HTTP-запросов к серверам Telegram и взаимодействия с ботами. Разработчики могут использовать различные языки программирования и библиотеки, чтобы взаимодействовать с Telegram Bot API и создавать ботов.

Telegram Bot API обеспечивает гибкость и функциональность для создания разнообразных ботов, включая информационные боты, сервисные боты, игровые боты и другие. Он также поддерживает функции аутентификации, шифрования и безопасности для обеспечения защиты данных и приватности пользователей.

В целом, Telegram Bot API предоставляет разработчикам мощный инструментарий для создания и взаимодействия с ботами в Telegram, открывая широкие возможности для автоматизации, коммуникации и предоставления услуг в этом популярной мессенджере.

Также, стоит отметить, что Telegram имеет криптографический протокол, который используется в системе обмена сообщениями для шифрования переписки пользователей – MTProto. Это собственный механизм, который обеспечивает работу этого мессенджера и который был разработан Павлом Дуровым и его командой разработчиков.

MTProto API, также известный как Telegram API, представляет собой программный интерфейс, через который ваше приложение Телеграм взаимодействует с сервером. Telegram API открыт для использования, что позволяет любому разработчику создать свой собственный клиент мессенджера.

На данный момент в Telegram существуют два инструмента API - Telegram Bot API и Telegram API. Первый предназначен для создания стандартных чат-ботов в Telegram, а второй позволяет разрабатывать полностью уникальные клиенты.

Для использования Bot API не требуется глубокое понимание протокола шифрования MTProto (см. схема 1). Вспомогательный сервер автоматически обрабатывает шифрование и устанавливает связь с Telegram API. Достаточно просто установить соединение с сервером через простой HTTPS-интерфейс, и он предоставит упрощенную версию Telegram API с простой разметкой и возможностью использования вебхуков.

Немногие знают, что боты могут работать напрямую через Telegram API. Более того, это позволяет обойти некоторые ограничения, которые нам предоставляет Bot API.

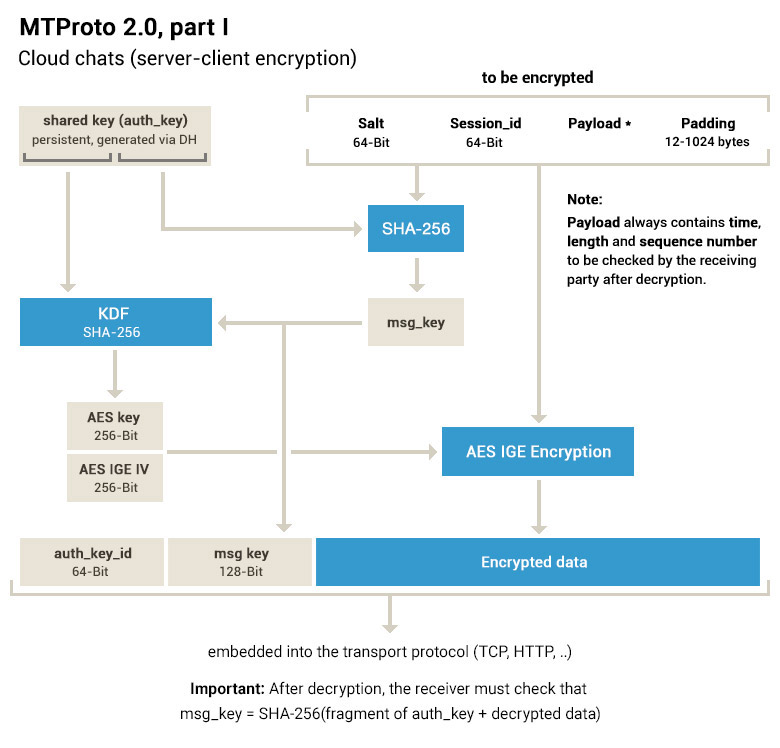


Схема 1. Схема работы алгоритма шифрования

Создание бота может показаться сложным и трудоемким процессом. Однако в Telegram существует платформа под названием Botfather, которая поможет даже новичкам справиться с этой задачей. Этот сервис облегчит получение прав доступа к уникальному боту, созданному самостоятельно. Процесс довольно прост. Botfather пошагово проведет вас через процесс создания токена, который необходим для связи бота с конструктором сайтов, где будут определены команды для настроенного бота.

Рассмотрим алгоритм действий по созданию бота в Telegram:

1. Зайдите в Telegram и найдите бота под названием @Botfather. Это официальный бот Telegram, который поможет вам создать и настроить своего собственного бота.
2. Начните диалог с Botfather, нажав на кнопку "Start" или отправив команду "/start".
3. Следуйте инструкциям Botfather. Он предложит вам различные команды и опции для настройки бота. Например, вы можете использовать команду "/newbot", чтобы создать нового бота.
4. При создании нового бота вам будет предложено выбрать имя для бота. Выберите уникальное имя, которое будет идентифицировать вашего бота.
5. После выбора имени Botfather предоставит вам токен доступа. Этот токен является ключом для связи вашего приложения или скрипта с ботом.
6. Сохраните токен доступа в безопасном месте, так как он будет использоваться для авторизации вашего бота.
7. Теперь ваш бот создан! Вы можете настроить его дальше, добавлять команды, ответы на сообщения и другую функциональность, в зависимости от ваших потребностей.
8. Для тестирования бота вы можете найти его в Telegram по имени, которое вы выбрали при создании, и начать с ним диалог.

Если искать информацию о создании Телеграм-бота на языке программирования Java, то вам порекомендуется обратить внимание на библиотеку TelegramBots, которая является популярным инструментом для разработки ботов на Java. Она предоставляет широкие возможности и удобный интерфейс для работы с Telegram API.

Также стоит отметить, что для более простых и начальных проектов можно использовать библиотеку JTelegramBotAPI, которая предоставляет базовый функционал для создания ботов на Java. Она может быть полезна, если вам нужно написать простого "hello-world" бота или ознакомиться с основами разработки ботов на данном языке.

Однако, если вы стремитесь к более продвинутой разработке и ищете библиотеку с расширенными возможностями, то рекомендуется обратить внимание на библиотеку TelegramBots. Она обладает асинхронной архитектурой, поддерживает использование декораторов и предоставляет удобные инструменты для разработки ботов на Java.

В целом, TelegramBots является наиболее популярным и рекомендуемым выбором для разработки Телеграм-ботов на языке программирования Java.

Ниже представлены Telegram боты, которые можно отнести к списку ботов, которые пользуются популярность в России [3]:

* @YandexTranslate – бот, который поможет вам перевести тексты с одного языка на другой. Он поддерживает множество языков и обеспечивает быстрые переводы (см. рисунок 1.1.1).
* @taxisniper\_bot – бот, который поможет вам рассчитать наиболее выгодную стоимость поездки на такси от различных перевозчиков. Просто укажите адрес и бот соберет необходимую информацию, чтобы вы могли выбрать наиболее оптимальный вариант (см. рисунок 1.1.2).
* @pogodarlingbot – бот, который предлагает рекомендации по одежде на конкретный день в указанном городе. Он учитывает данные о температуре, возможных осадках и других факторах, чтобы подсказать, какой гардероб будет наиболее комфортным в определенный момент времени (см. рисунок 1.1.3).

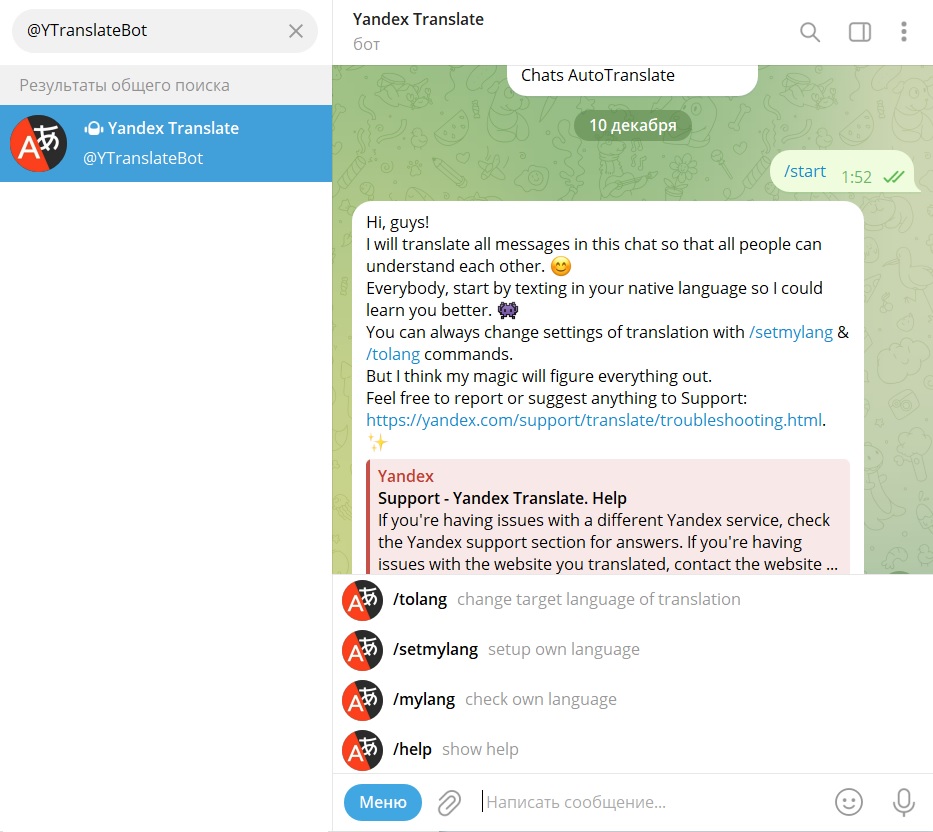


Рисунок 1.1.1

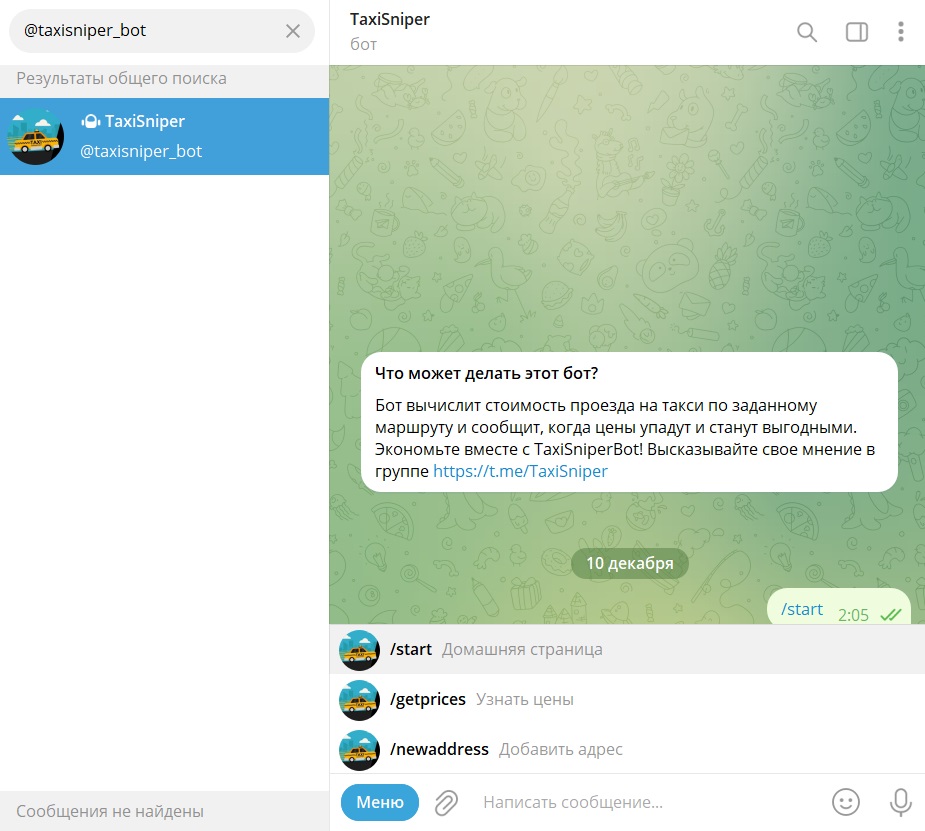


Рисунок 1.1.2

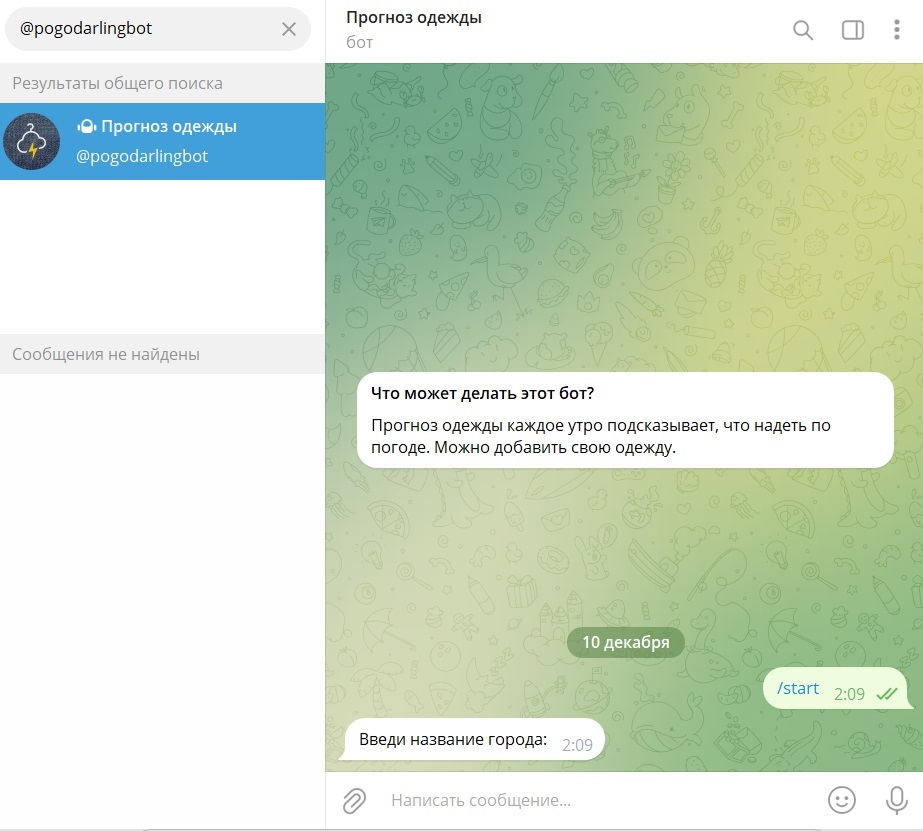


Рисунок 1.1.3

# Глава 2 Выбор технологий и среды разработки

# 2.1 Обоснование выбора Java как языка программирования

Существует обширное множество языков программирования, разработанных для решения различных задач. Каждый из них отличается уникальным набором операторов и определенным синтаксисом, но есть среди большого множества языков – Java. Стоит сразу дать точное определение, а именно 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 [4, 5].

Java является одним из самых популярных языков программирования по ряду причин, укажем основные:

1. Java обладает огромным и активным сообществом разработчиков по всему миру. Что всегда можно нам позволяет найти помощь, поддержку и обмен опытом со стороны опытных программистов. Сообщество Java активно участвует в разработке библиотек, фреймворков и инструментов, способствуя постоянному развитию экосистемы Java.
2. Данный язык имеет обширную и качественную документацию, которая предоставляет подробные сведения о языке, его возможностях, стандартных библиотеках и инструментах. Официальная документация Java, а также различные учебники, руководства и онлайн-ресурсы обеспечивают программистов всей необходимой информацией для разработки и отладки проектов на Java.
3. Java имеет огромное количество готовых решений в виде библиотек, фреймворков и инструментов, которые облегчают разработку и ускоряют процесс создания программного обеспечения. К примеру, в дальнейшем в своей работе для разработки бота для Telegram на Java доступны различные библиотеки, но мы используем TelegramBots API, который предоставляет высокоуровневые абстракции и упрощают взаимодействие с Telegram Bot API.
4. Язык Java является кроссплатформенным языком программирования, что означает, что код, написанный на Java, может выполняться на различных операционных системах. Это позволяет создать бота для Telegram на Java, который будет работать на различных платформах, включая Windows, macOS и Linux, без необходимости внесения значительных изменений в код.
5. Данный язык программирования широко используется в различных областях разработки, включая веб-разработку, мобильную разработку, разработку приложений настольного класса, игровую разработку и многое другое. Java имеет хорошую поддержку для работы с сетевыми протоколами, базами данных и веб-сервисами, что делает его подходящим выбором для разработки бота для Telegram с богатым функционалом

Стоит обязательно сказать, что Java сочетает в себе мощный язык программирования, богатую экосистему инструментов, удобных библиотек и фреймворков для работы с Telegram Bot API.

TelegramBots API одна из наиболее популярных библиотек для разработки ботов для Telegram на Java. Как мы говорили раннее она упрощает взаимодействие с Telegram Bot API и данная библиотека позволяет легко создавать и настраивать ботов, обрабатывать входящие сообщения, отправлять сообщения в чаты, работать с клавиатурами и многое другое. TelegramBots API обеспечивает удобный и интуитивно понятный способ разработки ботов для Telegram на Java. По сути Telegram Bot API сам по себе предоставляет широкий набор возможностей для разработки ботов, при этом Java обеспечивает удобный способ взаимодействия с этим API благодаря своей гибкости и инструментам, таким как работа с HTTP-запросами, сериализации и десериализации данных, обработке JSON и другим функциональным особенностям, которые могут быть использованы для взаимодействия.

Говоря о создании продукта на Java, нельзя не остановится более подробно на вопросах о строгой системе безопасности, которая построена на нескольких ключевых аспектах. Поговорим о них подробнее.

Одной из основных особенностей Java является то, что код компилируется в промежуточный байт-код, который выполняется на JVM («Java Virtual Machine»). JVM обеспечивает изоляцию кода и предоставляет механизмы безопасности, такие как проверка границ массивов, контроль типов, автоматическое управление памятью и многое другое. Благодаря этому, многие опасные операции, которые могут привести к уязвимостям, предотвращаются или обнаруживаются на этапе выполнения кода. Используется модель безопасности, основанная на ограничении прав доступа. Каждый класс или пакет имеет определенные права доступа, которые определяют, какой код может обращаться к его членам (полям, методам). Это позволяет контролировать доступ к чувствительным ресурсам и предотвращать несанкционированный доступ к ним. Также, Java имеет строгую систему проверки типов, которая помогает предотвратить множество ошибок и уязвимостей. Компилятор Java проверяет типы переменных и операций на этапе компиляции, что помогает предотвратить ошибки времени выполнения, такие как переполнение буфера или некорректное обращение к памяти. Java обеспечивает механизмы безопасности на уровне классов и пакетов. Классы и пакеты могут быть объявлены как final или sealed, что предотвращает их изменение или наследование другими классами. Это помогает защитить код от несанкционированных изменений. И стоит добавить, что Java имеет механизм обработки исключений, который позволяет предотвратить сбои приложения и обработать ошибки безопасным образом. Обработка исключений позволяет программистам обнаруживать и обрабатывать исключительные ситуации, такие как ошибки ввода-вывода или сетевые проблемы, и принимать соответствующие меры, чтобы приложение продолжило работу без вреда для безопасности.

Так как результатом данной работы выступает бот, созданный на Java, то нужно сказать, что он может быть более надежным и защищенным от уязвимостей благодаря строгой системе безопасности языка, а именно системе типов, проверке границ массивов, ограничения доступа и механизмам обработки исключений – все это помогают предотвратить ошибки, уязвимости и потенциальные атаки на приложение, что делает Java привлекательным выбором для создания безопасных и надежных ботов для Telegram.

# 2.2 Сравнение и выбор Spring boot как фреймворка для разработки веб-приложений

Прежде всего, необходимо отметить, что множество веб-фреймворков можно условно разделить на две большие группы:

1. Фронтенд-фреймворки — это инструменты, которые помогают разработчикам создавать интерактивные веб-страницы. Они включают в себя библиотеки и инструменты, которые упрощают разработку пользовательского интерфейса (UI) и пользовательского опыта (UX). Некоторые популярные фронтенд-фреймворки включают React, Angular, Vue и другие.

2. Бекенд-фреймворки — это программная платформа, которая помогает разработчикам создавать серверную часть веб-приложений. Она предоставляет инструменты для работы с базами данных, обработки запросов, маршрутизации, авторизации и других функций. Некоторые популярные бекенд-фреймворки включают Django, Ruby on Rails, Flask и другие.

Для проведения сравнительного анализа рассмотрим наиболее популярные в настоящее время бекенд-фреймворки более подробно (см. таблицу 2.2.1).

Таблица 2.2.1 – Современные бекенд-фреймворки [6]

|  |  |  |
| --- | --- | --- |
| *Название* | *Язык* | *Приложения* |
| Django | Python | National Geographic, Mozilla, Pinterest |
| Laravel | PHP | Deltanet Travel, Neighborhood Lender, World Walking |
| Ruby on Rails | Ruby | Twitter, Zendesk, Github |
| Express JS | Node | Uber, Groupon, GoDaddy |

К особенностям Django относится высокая производительность, ORM, шаблонизатор, система управления пользователями, поддержка многих функций. Используется архитектурный шаблон MTV, который позволяет разработчикам быстро создавать веб-приложения с высокой производительностью. Имеется встроенный ORM, который позволяет работать с базами данных более эффективно. Предлагается мощный шаблонизатор, который позволяет генерировать HTML-страницы быстро и эффективно. Имеется мощная система управления пользователями, которая позволяет разработчикам легко управлять доступом к ресурсам приложения. Django поддерживает множество других функций, которые могут быть полезны для создания полнофункциональных веб-приложений. Давайте поговорим о недостатках Django. Например, он может быть сложен для новичков, так как требует определенных знаний и навыков для работы с ним. Кроме того, Django может быть не самым оптимальным выбором для небольших проектов, так как его использование может быть избыточным.

Laravel преимущественен тем, что он достаточно простой в использовании, у него богатая экосистема, безопасный, масштабируемый, производительный. Имеется простой и понятный синтаксис, который упрощает разработку приложений. Предлагается большая экосистема пакетов и расширений, которые можно использовать для расширения функциональности приложения. Обеспечивается высокий уровень безопасности для приложений, благодаря встроенным функциям защиты от SQL-инъекций и XSS-атак. Есть возможность легко масштабировать приложения, благодаря поддержке множества баз данных и протоколов связи. Laravel оптимизирован для высокой производительности, что позволяет создавать быстрые и надежные приложения. Но также есть У Laravel и свои недостатки. Один из них - сложность в настройке. Laravel требует определенной настройки перед началом работы, и это может быть сложно для новичков. Кроме того, он может быть менее гибким, чем некоторые другие фреймворки, например такие как Ruby on Rails.

Особенности Ruby on Rails (RoR) относят простоту опять же в использовании, большую экосистему, высокую производительность, гибкость, открытый исходный код. Также, имеется простой и понятный DSL (Domain-Specific Language), который упрощает процесс разработки. Он также предлагает готовые решения для многих задач, что сокращает время на разработку. Предоставляется огромная экосистема пакетов, расширений и библиотек, которые можно использовать в своих проектах. Это позволяет разработчикам находить готовые решения для своих задач и экономить время на разработке. Фреймворк оптимизирован для обеспечения высокой производительности приложений. Он использует асинхронное выполнение и другие методы для повышения производительности. Rails позволяет разработчикам быть очень гибкими в своих решениях. Поддерживает различные базы данных, протоколы и технологии, что дает возможность создавать приложения с максимальной гибкостью. Отличается открытым исходным кодом и имеет активное сообщество разработчиков, которые постоянно улучшают и развивают его. Это обеспечивает постоянное улучшение качества и возможностей фреймворка. RoR есть свои недостатки, такие как cложность изучения, зависимость от Ruby, ограниченная поддержка других языков. Rails имеет сложный DSL и множество концепций, которые могут быть трудными для понимания новыми разработчиками. Работает только на языке программирования Ruby, что ограничивает его использование только теми, кто знает этот язык. Хотя Rails поддерживает несколько других языков, таких как JavaScript и CoffeeScript, они не так широко используются, как в других фреймворках.

Express JS уникален своей производительностью, также прост, масштабируем, но при этом большая поддержка сообщества. Express.js был разработан с учетом высокой производительности, что делает его идеальным для крупномасштабных приложений. Имеет простой и понятный API, с которым легко начать работу. Поддерживает широкий спектр функций, что делает его подходящим как для небольших, так и для крупных проектов. Существует большое и активное сообщество разработчиков, которые вносят свой вклад в проект, что гарантирует всегда доступную поддержку. Хотя Express.js обладает многими преимуществами, у него есть и некоторые недостатки. Express.js не предлагает такие функции, как автоматическая миграция базы данных или встроенная поддержка ORM. Эти функции необходимо добавлять вручную или использовать дополнительные библиотеки. Настройка Express.js может быть сложной задачей для новичков, поскольку требует понимания Node.js и его модулей.

Итак, почему именно Spring Boot оправдывает все поставленные задачи? Одними из главных особенностей и преимуществ являются быстрая разработка, поддержка различных технологий, минимальное количество кода, масштабируемость, безопасность, производительность. Spring Boot предлагает автоматическую конфигурацию, что ускоряет процесс разработки. Поддерживаются различные технологии, такие как Spring MVC, Spring Security и др., это позволяет разрабатывать веб-приложения различной сложности. Присутствует возможность создавать веб-приложения с минимальным количеством кода, что ускоряет разработку и уменьшает вероятность ошибок.

Spring Boot поддерживает масштабирование приложений на разных уровнях, от разработки до промышленной эксплуатации. Имеется встроенные механизмы безопасности, которые помогают защитить приложение от атак. Spring Boot оптимизирован для обеспечения высокой производительности веб-приложений.

Таким образом, исходя из перечисленных выше преимуществ, Spring Boot является оптимальным выбором для разработки Telegram-бота на языке программирования Java, поскольку обеспечивается быстрая разработка, поддержка различных технологий, минимальное количество кода, масштабируемость, безопасность, производительность.

# 2.3 Выбор и настройка PostgreSQL как базы данных для хранения информации о пользователях и бронированиях

PostgreSQL — это система управления базами данных (СУБД), которая является свободной и открытой. Она была создана в 1980-х годах и используется во многих приложениях, включая веб-сайты, мобильные приложения и другие. PostgreSQL поддерживает множество функций, включая транзакции, индексы, триггеры и многое другое. Она также имеет высокую производительность и надежность, что делает ее популярной среди разработчиков [7]. PostgreSQL имеет высокую производительность и масштабируемость, возможность гибкой настройки, поддержку различных типов данных, является безопасной. Данный СУБД может обрабатывать большие объемы данных и может масштабироваться для обработки миллионов запросов в секунду. Также он позволяет пользователям настраивать многие аспекты работы базы данных, что может быть полезно для оптимизации производительности. Поддерживается большое количество типов данных, включая текстовые, числовые, географические и другие, что делает его универсальным решением для различных задач.

PostgreSQL является открытым программным обеспечением, что означает, что его можно использовать бесплатно и изменять в соответствии с потребностями пользователя. Имеются встроенные механизмы защиты от различных видов атак, таких как SQL-инъекции и другие виды атак на базу данных. PostgreSQL имеет активное сообщество разработчиков и пользователей, которые постоянно работают над улучшением и развитием этой СУБД.

Как и любая другая СУБД, PostgreSQL имеет свои недостатки. В частности, он сложен настройкой и администрированием, требует большое количество ресурсов, отсутствуют некоторые функции, такие как хранение процедур и триггеров, которые могут быть необходимы для некоторых приложений. PostgreSQL может быть сложным для настройки и администрирования для новичков и нетехнических пользователей. Требуется больше ресурсов, таких как память и процессор, по сравнению с некоторыми другими СУБД, такими как MySQL.

В целом, настройка базы данных PostgreSQL включает в себя несколько этапов, в зависимости от того, используется ли существующая база данных или создается новая. Ниже перечислены основные шаги, которые могут помочь настроить PostgreSQL [8]:

1. Установка PostgreSQL на целевой сервер или компьютер.
2. Создание новой базы данных, используя команду CREATE DATABASE в SQL-запросе.
3. Подключение к созданной базе данных, используя клиент PostgreSQL, такой как pgAdmin.
4. Импорт данных в базу данных, используя команды INSERT, UPDATE и DELETE в SQL-запросах.
5. Настройка прав доступа и разрешений для пользователей и ролей в базе данных, используя систему управления доступом PostgreSQL.
6. Оптимизация производительности базы данных через настройку параметров конфигурации PostgreSQL, такие как параметры буфера, параметры ввода-вывода и параметры памяти.
7. Регулярное создание резервных копий базы данных для восстановления после сбоев или для переноса на другой сервер.

База данных предложенного в данной работе бота включает четыре таблицы: bookings, booking\_objects, types, users. Каждая из них связана с отдельной сущностью, определённой в отдельном классе, и отражает часть предметной области исходной задачи. Далее необходимо рассмотреть каждую из перечисленных выше таблиц подробнее.

Таблица bookings содержит информацию о бронированиях и включает в себя поля, представленные в таблице 2.3.1.

Таблица 2.3.1 – Структура таблицы bookings

|  |  |  |
| --- | --- | --- |
| *Тип поля* | *Название поля* | *Значение поля* |
| Long | id | Идентификатор |
| LocalDateTime | timeStart | Время начала |
| LocalDateTime | timeEnd | Время окончания |
| boolean | isApproved | Статус подтверждения |
| User | user | Пользователь |
| BookingObject | bookingObject | Объект бронирования |

Таблица booking\_objects содержит информацию об объектах бронирования и включает в себя поля, представленные в таблице 2.3.2.

Таблица 2.3.2 – Структура таблицы booking\_objects

|  |  |  |
| --- | --- | --- |
| *Тип поля* | *Название поля* | *Значение поля* |
| Long | id | Идентификатор |
| String | name | Название |
| String | description | Описание |
| String | image | Изображение |
| LocalTime | availableFrom | Время открытия доступа |
| LocalTime | availableTo | Время закрытия доступа |
| Type | type | Тип бронирования |

Таблица types содержит информацию о типах бронирования и включает в себя поля, представленные в таблице 2.3.3.

Таблица 2.3.3 – Структура таблицы types

|  |  |  |
| --- | --- | --- |
| *Тип поля* | *Название поля* | *Значение поля* |
| Long | id | Идентификатор |
| String | name | Наименование |

Таблица users содержит информацию о пользователях и включает в себя поля, представленные в таблице 2.3.4.

Таблица 2.3.4 – Структура таблицы users

|  |  |  |
| --- | --- | --- |
| *Тип поля* | *Название поля* | *Значение поля* |
| Long | id | Идентификатор |
| String | username | Имя пользователя |
| boolean | isAdmin | Наличие админстатуса |
| List<Booking> | bookings | Список бронирований |

Таким образом, представленная выше структура базы данных разработанного Telegram-бота позволяет оптимально организовать работу с набором минимально необходимых данных для осуществления целевой сервисной деятельности. Эта база данных позволяет осуществлять все основные CRUD-операции: создание, чтение, обновление и удаление.

Дальнейшее совершенствование предложенной базы данных возможно по следующим направлениям:

1. Улучшение производительности. Оптимизация SQL-запросов, использование индексов, настройка параметров конфигурации.

2. Добавление новых функций. Поддержка различных видов бронирования, интеграция с другими системами, улучшение пользовательского интерфейса.

3. Обеспечение безопасности. Защита от атак, таких как SQL-инъекции, подбор паролей и другие виды вредоносного кода.

4. Улучшение масштабируемости. Добавление возможностей горизонтального масштабирования для поддержки большего количества пользователей и транзакций.

5. Повышение надежности. Обеспечение отказоустойчивости системы, резервирование данных и восстановление после сбоев.

# Глава 3 Разработка телеграмм-бота для бронирования

# 3.1 Описание функциональности и интерфейса бота

Для начала работы с Telegram-ботом необходимо перейти на его страницу в программе. Для запуска бота предусмотрена стандартная команда «/start» или кнопка «ЗАПУСТИТЬ», при отправке, в первом случае или нажатии, во втором, появляется приветственное сообщение и инициализируется процесс определения дальнейших команд (см. рисунок 3.1.1). При запуске бота пользователем определяются две основные команды.



Рисунок 3.1.1

Интерфейс начальной страницы содержит две основные кнопки выбора: «Забронировать» (где пользователь выбирает категорию для дальнейшей брони) и «Мои бронирования» (где пользователь бота может ознакомится с перечнем текущих его броней), реализующие основной функционал бота. При нажатии на кнопку «Забронировать», осуществляется переход на страницу бронирования (см. рисунок 3.1.2). На данной странице пользователь выбирает категорию бронирования: фитнес, аренда сноуборда, аренда машины.

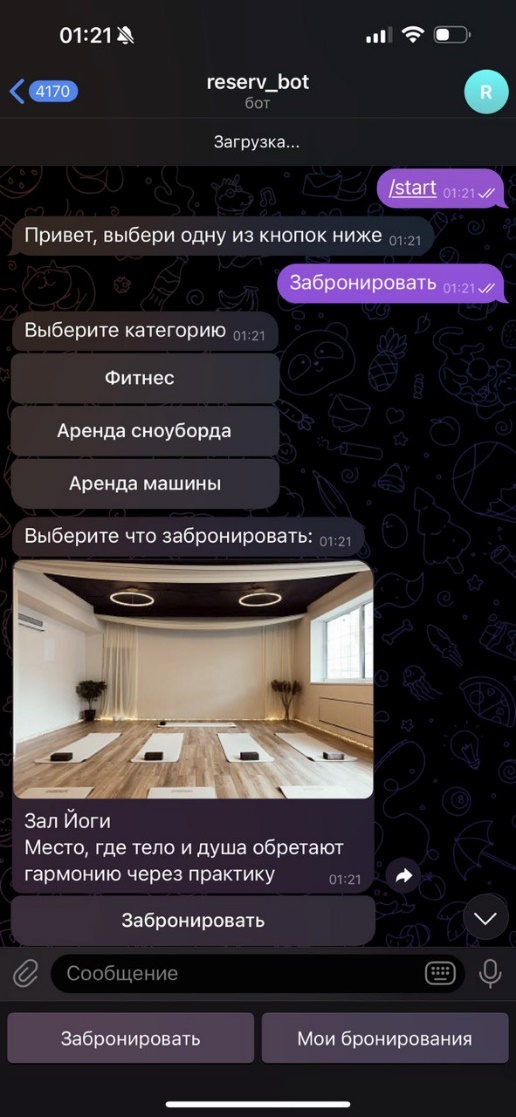


Рисунок 3.1.2

Далее необходимо выбрать конкретный раздел для бронирования в следствии чего указать время начала и завершения бронируемого периода (см. рисунок 3.1.3). При успешном бронировании появится соответствующее сообщение.

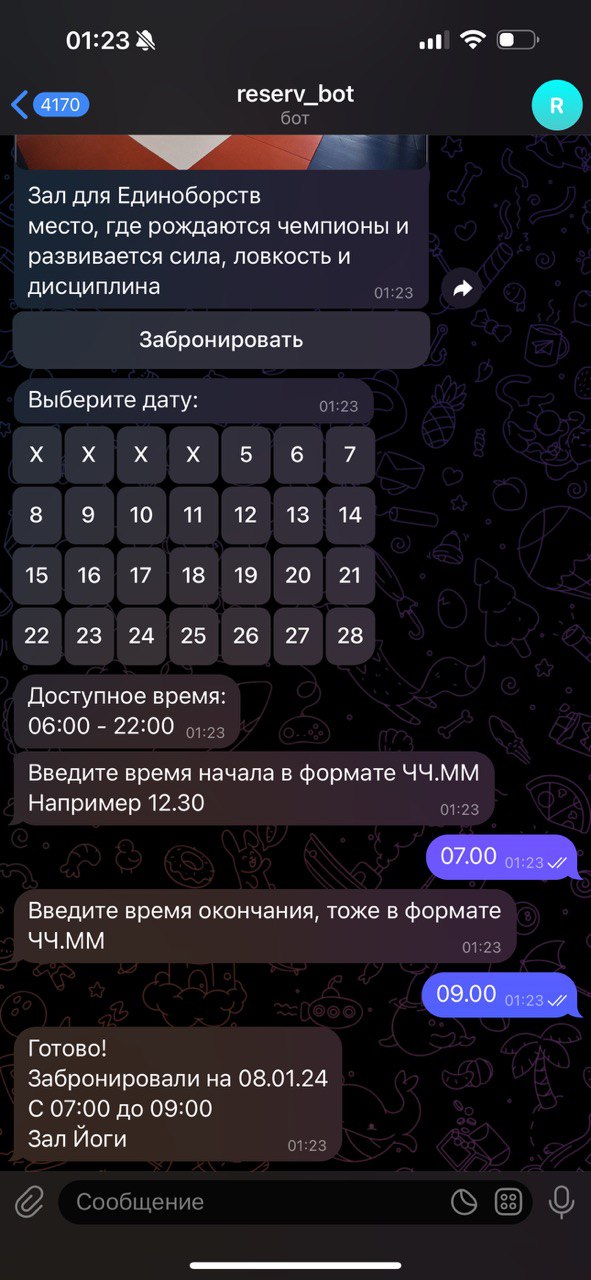


Рисунок 3.1.3

Для более объективной картины со стороны пользователя изобразим UseCase диаграмму (см. рисунок 3.1.4). Напомним, что диаграмма использования (Use Case diagram) является графическим представлением функциональности системы и отображает взаимодействие актеров (пользователей или внешних систем) с системой для достижения конкретных целей. Она помогает визуализировать основные функциональные возможности системы и ее взаимодействие с внешним миром [9].

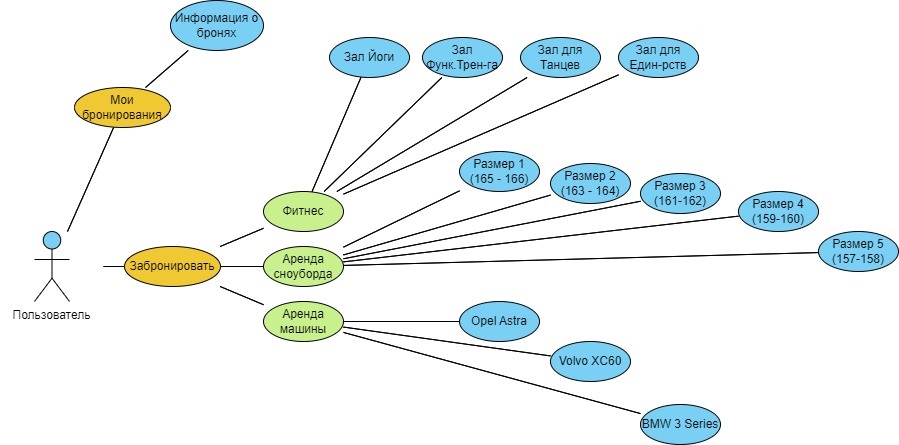


Рисунок 3.1.4

Эта диаграмма показывает, как пользователь может забронировать разные виды услуг с помощью данного телеграм-бота. На диаграмме есть 3 категории:

* Фитнес
* Аренда сноуборда
* Аренда машины

Для каждой категории есть несколько вариантов, из которых пользователь может выбрать. Например, для раздела «фитнес» можно выбрать зал йоги, зал функциональных тренировок, танцевальный зал или единоборства. Для аренды сноуборда можно выбрать от 1 до 5 размера, в зависимости от роста пользователя. Для аренды машины можно выбрать между автомобилями Opel Astra, Volvo XC60 или BMW 3 Series.

Также нельзя не отметить, что пользователю предоставляется возможность просмотра забронированного зала, сноубордической доски или автомобиля (см. рисунок 3.1.5).

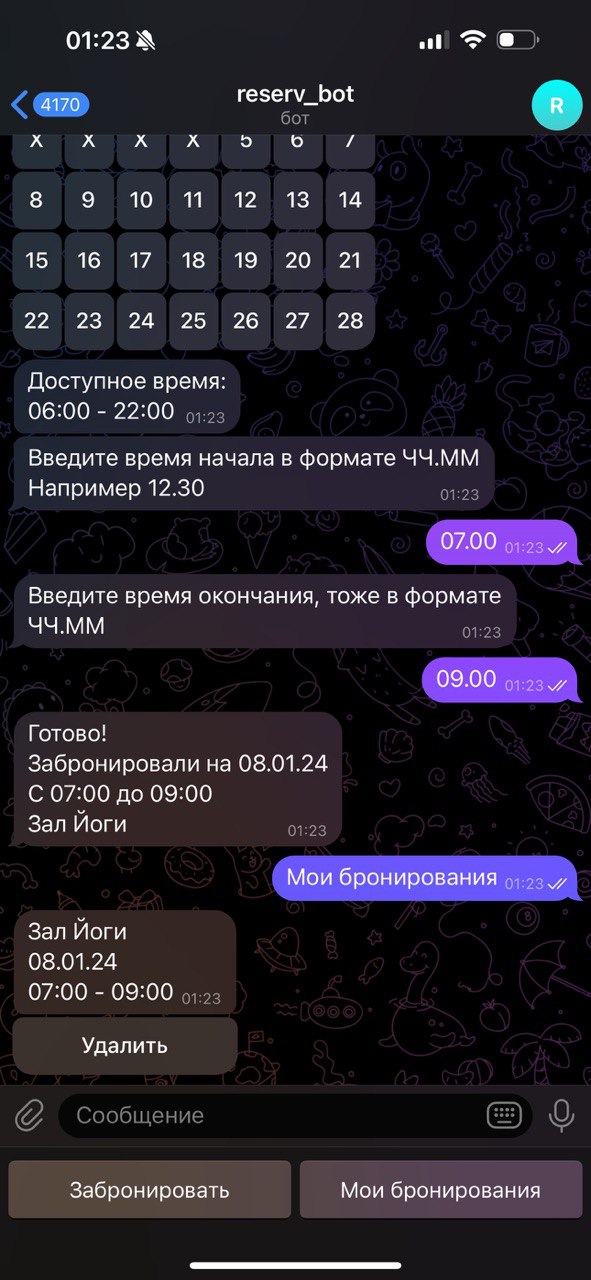


Рисунок 3.1.5

Рассмотрим, администраторский интерфейс, который отличается наличием двух дополнительных кнопок, а именно:

- Добавить объект

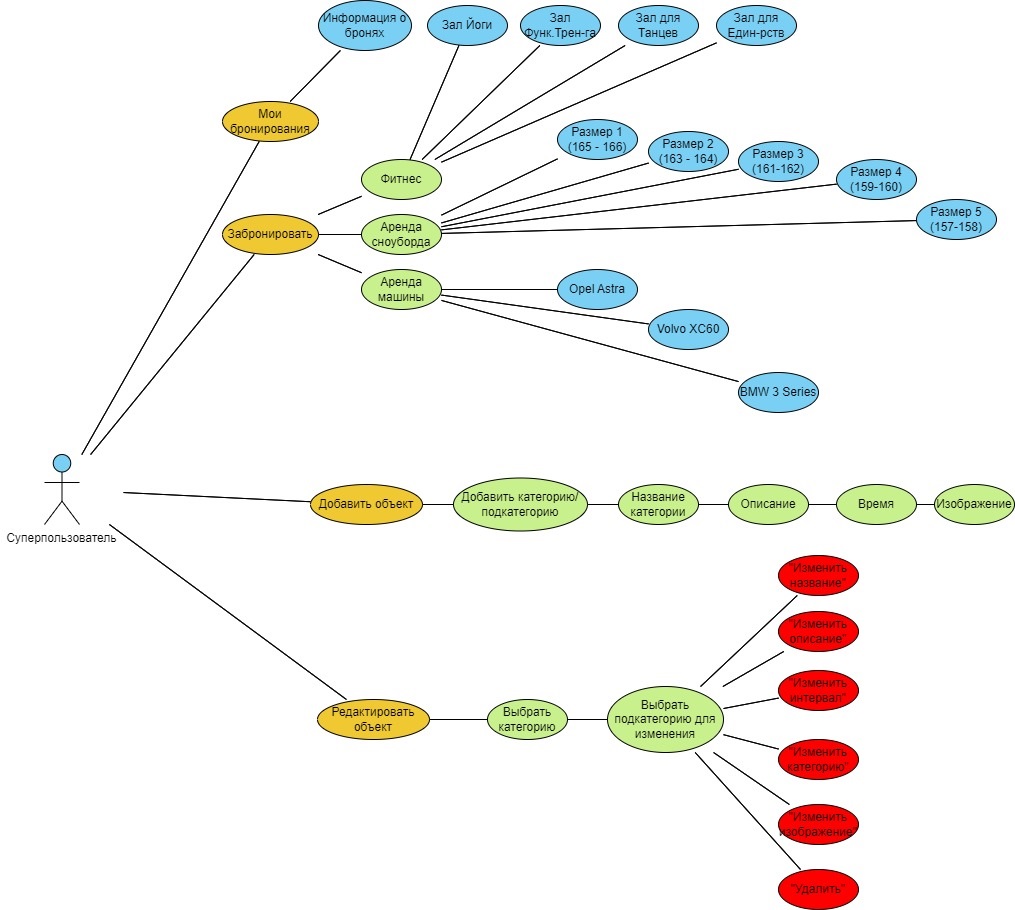
- Редактировать объект

«Объектом», в рамках заданной кнопки, мы считаем любую категорию или подкатегорию брони. Другими словами, чтобы задать объект, администратор должен добавить ту или иную «категорию» для дальнейшего внесения объекта брони (см. рисунок 3.1.6).



см. рисунок 3.1.6

Снова обратимся к UseCase диаграмме (см. рисунок 3.1.8), но уже для визуализации основного функциональна возможностей администратора данного Telegram бота.



Таким образом, разработанный в данном проекте Telegram-бот обладает, в целом, стандартным интерфейсом, создание которого обусловлено и ограничено внешним API. Дальнейшее совершенствование рассматриваемого бота возможно по двум основным направлениям: расширение функционала – добавление новых и улучшение существующих функций, кастомизация интерфейса. В Telegram есть возможность кастомизации интерфейса для пользователей с ограниченными возможностями. Это включает в себя изменение размера шрифта, выбора темы и настройки контрастности. Кроме того, Telegram предлагает поддержку различных языков и возможность использовать клавиатуру для отправки сообщений.

# 3.2 Реализация основных методов и классов бота

Структурно разработанный Telegram-бот включает один пакет – reserve.bot, который содержит начальный класс ReserveBot2Application.java, запускающий Spring Boot, и пять директорий с отдельными классами: bot (см. приложение А), command, models, repository, service. Далее рассмотрим каждую из перечисленных групп методов подробнее.

Директория bot содержит пять классов и один интерфейс BotService.java (см. таблица 3.2.1).

Таблица 3.2.1 – Структура директории bot

|  |  |
| --- | --- |
| *Класс (интерфейс)* | *Методы* |
| сlass Bot | Методы, определяющие действия при обновлении данных, и возвращающие параметры бота |
| interface BotService | Методы отправки текста, разметки, фото, методы загрузки фото |
| class BotServiceImpl | Реализация методов, указанных в интерфейсе BotService |
| class Button | Конструктор кнопки |
| class Runner | Метод регистрации бота |
| class UpdateUtil | Методы получения идентификатора пользователя и результата ввода данных |

Рассмотренные в таблице 3.2.1 классы и методы обеспечивают общую функциональность Telegram-бота и привязку к Telegram API, предоставляющему возможности выполнять различные задачи, такие как отправка сообщений, создание групп и каналов, управление файлами и многое другое.

Директория command содержит три класса, один интерфейс и поддиректорию, классы которой реализуют данный интерфейс: AddObject.java, EditObject.java, MyBookings.java, NewBookings.java, Start.java (см.таблицу 3.2.2).

Таблица 3.2.2 – Структура директории command

|  |  |
| --- | --- |
| *Класс (интерфейс)* | *Методы* |
| interface Command | Метод, предопределяющий исполнение команды |
| class CommandContainer | Конструктор контейнера, методы возвращающие команду и факт её наличия |
| enum CommandName | Содержит перечень доступных команд |
| class KeyboardMaker | Методы, обеспечивающие переключение режимов клавиатуры |

Рассмотренные в таблице 3.2.2 классы и методы определяют команды, которые пользователь и администратор бота может использовать для управления работой программы.

Директории models и repository предназначены для создания сущностей на основе классов для оптимизации работы с базой данных PostgreSQL, которую мы рассматривали в пункте 2.3 данной дипломной работы.

Директория models содержит следующие классы: Booking.java, BookingObjects.java, Type.java, User.java.

Директория repository содержит следующие классы: BookingRepository.java, BookingObjectsRepository.java, TypeRepository.java, UserRepository.java.

Последняя директория service содержит только четыре класса (см. таблицу 3.2.3).

Таблица 3.2.2 – Структура директории service

|  |  |
| --- | --- |
| *Класс (интерфейс)* | *Методы* |
| class BookingObjectService | Методы поиска, сохранения и удаления объекта бронирования (бронирования) |
| class BookingService |
| class TypeService | Методы поиска и сохранения типа бронирования |
| class UserService | Методы поиска и создания пользователя |

Таким образом, реализация основных методов и классов разработанного Telegram-бота позволяет использовать Telegram API и модель ORM для функционирования данной программы бронирования. Эта модель позволяет работать с данными в объектном формате, вместо того чтобы использовать SQL-запросы напрямую. Это помогает разработчикам сосредоточиться на логике приложения, а не на работе с базой данных. Некоторые преимущества использования ORM включают в себя ускорение разработки, уменьшение количества кода и улучшение производительности.

Однако, использование ORM также может привести к некоторым недостаткам, таким как снижение производительности в некоторых случаях и необходимость более тщательного проектирования базы данных. В целом, выбор между использованием ORM или написанием SQL-запросов напрямую зависит от конкретных потребностей проекта и перспектив его дальнейшего развития.

# 3.3 Тестирование и отладка бота

Для тестирования разработанного Telegram-бота использовались следующие виды тестирования [10, 11]:

1. Функциональное тестирование проводилось путём проверки основных функций бота: создание бронирования, просмотр бронирований, добавление объекта бронирования, редактирование объекта бронирования. Результат тестирования показал, что все функции работают в соответствии с их предназначением.

2. Валидация входных данных проводилась путём ввода различных комбинаций входных данных, включая те, ввод которых не предусмотрен логикой программы. Использовались пробелы, буквы разного регистра, цифры и другие символы. Результат тестирования показал, что ввод некорректных данных отрабатывается штатными средствами программы.

3. Тестирование пользовательского интерфейса проводилось с привлечением внешних пользователей. Результаты опроса целевой аудитории по тестированию пользовательского интерфейса представлены в таблице 3.3.1.

Таблица 3.3.1 – Результаты опроса по тестированию UI бота

|  |  |
| --- | --- |
| Варианты ответов | Количество респондентов |
| Интерфейс не требует доработки | 34% |
| Необходима мелкая доработка элементов интерфейса | 29% |
| Необходима существенная доработка элементов интерфейса | 10% |
| Необходимо добавление новых элементов интерфейса | 27% |

4. Тестирование совместимости проводилось путём запуска разработанного Telegram-бота на различных типах устройств, включая персональные компьютеры и смартфоны различных производителей, и операционных систем (см. таблицу 3.3.2). Тестирование совместимости не проводилось на планшетах и смарт-часах, а также некоторых других менее распространённых устройствах.

Таблица 3.3.2 – Результаты теста совместимости

|  |  |  |
| --- | --- | --- |
| Устройство | ОС | Результат теста |
| Компьютер | Windows | Пройден |
| Mac OS | Пройден |
| Linux | Пройден |
| Смартфон | iOS | Пройден |
| Android | Пройден |

5. Тестирование производительности проводилось путём хронометража ответов бота и обработки пользовательских действий. Результат тестирования показал, что время работы программных функций соответствует среднему для Telegram-ботов подобного класса.

6. Интеграционное тестирование проводилось путём оценки программного взаимодействия разработанного Telegram-бота с Telegram API и базой данных PostgreSQL. Результат теста показал, что бот успешно работает с указанными программными элементами в интерактивном режиме.

Для модульного тестирования на основе фреймворка Junit в папке src был создан пакет test c тестовым классом BotTest.java.

Листинг 3.3.1 – Код тестового класса BotTest.java

|  |
| --- |
| import org.junit.jupiter.api.Test;  import static org.junit.jupiter.api.Assertions.\*;  import org.telegram.telegrambots.bots.TelegramLongPollingBot;  import reserve.bot.bot.Bot;  import reserve.bot.command.CommandContainer;  import static org.mockito.Mockito.mock;  public class BotTest {  @Test  public void testBotClass() {  Bot bot = new Bot();  assertNotNull(bot.getBotUsername());  assertNotNull(bot.getBotToken());  assertNotNull(bot.getCurrentUserCommands());  assertTrue(bot instanceof TelegramLongPollingBot);  CommandContainer commandContainer = mock(CommandContainer.class);  bot.setCommandContainer(commandContainer);  assertEquals(commandContainer, bot.getCommandContainer());  }  } |

Таким образом, были проведены следующие виды тестирования: функциональное тестирование, валидация входных данных, тестирование пользовательского интерфейса, тестирование совместимости, тестирование производительности, интеграционное тестирование, а также составлены unit-тесты. Результаты проведённых тестов позволили подтвердить соответствие разработанного Telegram-бота исходным требованиям и определить пути его дальнейшего совершенствования.

# 3.4 Документирование кода и работы бота

Документирование кода и работы приложения — это процесс создания документации, которая описывает структуру, функции и работу приложения. Это помогает другим специалистам понять, как работает приложение, и как они могут его использовать. Документирование также может включать описание того, как был создан код, какие технологии использовались, какие стандарты кодирования были использованы, и другие элементы [12].

Существует несколько способов документирования кода:

– создание комментариев в коде. это наиболее распространенный способ. комментарии должны описывать, что делает каждая функция или метод, какие параметры она принимает и какой результат возвращает.

– использование систем контроля версий, таких как git, для хранения истории изменений кода. это позволяет видеть, какие изменения были внесены в код, и как они повлияли на его работу.

– создание проектной документации. проектная документация может включать в себя описание архитектуры проекта, описание используемых технологий и стандартов кодирования, а также описание того, как взаимодействуют различные компоненты проекта.

– написание пользовательской документации. пользовательская документация может включать в себя инструкции по использованию приложения, примеры использования, часто задаваемые вопросы и т.д.

Основные правила документирования кода включают в себя: ясность и понятность, актуальность, полнота, структурность, единообразие. Документация должна быть написана так, чтобы ее можно было понять без дополнительных объяснений, отражать текущую версию кода и не содержать устаревших сведений, содержать всю необходимую информацию о коде, включая описание функций, параметров, возвращаемых значений и т.д. Документация должна иметь четкую структуру, которая облегчает ее чтение и понимание, использовать единый стиль написания и оформления, чтобы избежать путаницы.

В настоящее время стандартом документирования в Java-приложениях считается Javadoc. Он позволяет создавать HTML-документы, которые содержат информацию о классах, методах, полях и других элементах Java-кода. JavaDoc автоматически генерирует документацию на основе комментариев, которые написаны в коде. Для разметки комментариев Javadoc использует дескрипторы: @author, @version, @since, @see, @param, @return и другие.

Многие современные IDE поддерживают генерацию Javadoc. Например, в наиболее популярной среде Java-разработки IJ Idea этот процесс выглядит следующим образом: заходим в меню Tools выбираем пункт Javadoc и в появившемся окне определяем параметры автогенерации и сохранения.

Наиболее популярными инструментами, применяемыми при документировании кода, являются следующие [13]:

1. Jira - это система управления проектами, которая используется для организации работы над проектами. Она позволяет создавать задачи, назначать их на определенных исполнителей, контролировать сроки выполнения и т.д.

2. GitHub – это сервис для хранения кода и VCS. Он позволяет воспользоваться функцией wiki, которая позволяет создать, по сути, собственную базу знаний. Данный сервис облегчает совместную работу и доступ к документации по проекту.

3. GitBook - это платформа для создания электронных книг и документации. Она позволяет авторам создавать документы, добавлять в них текст, изображения, видео и другие материалы, а затем публиковать их в интернете. GitBook также предоставляет возможность совместной работы над документами, что позволяет командам работать над документами вместе.

4. Doxygen - это программа для генерации документации из комментариев в исходном коде программы. Она поддерживает множество языков программирования, включая C++, C#, Java, Python и другие. Doxygen позволяет создавать документацию в форматах HTML, LaTeX, RTF, PDF и других.

5. DokuWiki - это легковесная вики-система с открытым исходным кодом, написанная на PHP. Она предназначена для создания простых и мощных вики-систем, а также может использоваться для создания документации и других типов веб-сайтов. DokuWiki имеет простой и понятный синтаксис, который позволяет пользователям легко создавать и редактировать страницы.

6. Confluence — это система управления знаниями с веб-интерфейсом, которая помогает командам совместно работать над проектами и документами. Она позволяет пользователям создавать страницы, добавлять на них текст, картинки, видео, ссылки на другие страницы и документы, а также обсуждать и редактировать содержимое страниц. Confluence может быть интегрирована с другими приложениями и сервисами, такими как Jira, GitHub, Google Drive и другими, что позволяет пользователям работать с информацией более эффективно.

Таким образом, для документирования кода и работы предложенного в данной работе Telegram-бота необходимо создать комментарии в исходном коде классов, полей и методов данной программы. Стиль написания подобных комментариев должен соответствовать стандарту Javadoc, что позволит автоматически сгенерировать документацию. Также можно использовать отдельные инструменты документирования, рассмотренные в данной работе или их комбинацию.

# Заключение

Исходя из результатов рассмотрения вопросов, связанных с созданием Telegram-бота, в данной работе, можно сделать следующие выводы:

1. Spring Boot является оптимальным выбором для разработки Telegram-бота на языке программирования Java, поскольку обеспечивается быстрая разработка, поддержка различных технологий, минимальное количество кода, масштабируемость, безопасность, производительность.

2. Представленная в работе структура базы данных разработанного Telegram-бота позволяет оптимально организовать работу с набором минимально необходимых данных для осуществления целевой сервисной деятельности. Эта база данных позволяет осуществлять все основные CRUD-операции: создание, чтение, обновление и удаление.

3. Разработанный Telegram-бот обладает, в целом, стандартным интерфейсом, создание которого обусловлено и ограничено внешним API. Дальнейшее совершенствование рассматриваемого бота возможно по двум основным направлениям: расширение функционала – добавление новых и улучшение существующих функций, кастомизация интерфейса.

4. Реализация основных методов и классов разработанного Telegram-бота позволяет использовать Telegram API и модель ORM для функционирования данной программы бронирования. Эта модель позволяет работать с данными в объектном формате, вместо того чтобы использовать SQL-запросы напрямую.

5. Для оценки разработанного бота проводились следующие виды тестирования: функциональное тестирование, валидация входных данных, тестирование пользовательского интерфейса, тестирование совместимости, тестирование производительности, интеграционное тестирование, а также составлены unit-тесты. Результаты проведённых тестов позволили подтвердить соответствие разработанного Telegram-бота исходным требованиям и определить пути его дальнейшего совершенствования.

6. Для документирования кода и работы предложенного в данной работе Telegram-бота необходимо создать комментарии в исходном коде классов, полей и методов данной программы. Стиль написания подобных комментариев должен соответствовать стандарту Javadoc, что позволит автоматически сгенерировать документацию. Также можно использовать отдельные инструменты документирования, рассмотренные в данной работе или их комбинацию

# Список использованных источников

1. Telegram // <https://ru.wikipedia.org/wiki/Telegram>
2. Telegram Bot API // https://core.telegram.org/bots/api
3. Боты в Telegram: для чего нужны, как создать // <https://gb.ru/blog/boty-v-telegram/>
4. Центр справки Java // <https://www.java.com/ru/download/help/index.html>
5. Java // <https://ru.wikipedia.org/wiki/Java>
6. София Иванова. 10 Лучших Фреймворков для Фронтенда и Бэкенда // back4app - https://blog.back4app.com/ru/10-лучших-фреймворков-для-фронтенда-и-бэ/
7. What is PostgreSQL? // postgresql.org - https://www.postgresql.org/about
8. Настройка PostgreSQL под Linux // Хабр. - <https://habr.com/ru/companies/lsfusion/articles/590599>
9. Visual Paradigm // https://online.visual-paradigm.com/
10. Фундаментальная теория тестирования // Хабр. - <https://habr.com/ru/articles/549054>
11. Как реализовать end-to-end-тестирование telegram-бота // Хабр. - <https://habr.com/ru/companies/beeline_cloud/articles/734164>
12. Документирование кодовой базы. Зачем и как? // Хпбр. - <https://habr.com/ru/articles/565342>
13. 10 лучших инструментов документирования свободного программного обеспечения// GeeksForGeeks. - <https://www.geeksforgeeks.org/top-10-free-software-documentation-tools>

# Приложение

**Приложение А**

Листинг основных классов Telegram-бота

**Класс Bot.java**

package reserve.bot.bot;

import lombok.RequiredArgsConstructor;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

import org.telegram.telegrambots.bots.TelegramLongPollingBot;

import org.telegram.telegrambots.meta.api.objects.Update;

import reserve.bot.command.Command;

import reserve.bot.command.CommandContainer;

import java.util.HashMap;

import java.util.Map;

@RequiredArgsConstructor

@Component

public class Bot extends TelegramLongPollingBot {

@Value("${bot.username}")

private String botUsername;

@Value("${bot.token}")

private String botToken;

private final CommandContainer commandContainer;

private final Map<Long, Command> currentUserCommands = new HashMap<>();

@Override

public void onUpdateReceived(Update update) {

Long userId = UpdateUtil.getUserId(update);

String input = UpdateUtil.getInput(update);

boolean isFinished = false;

if (commandContainer.hasCommand(input)) { // user starts new command

Command command = commandContainer.getCommand(input);

currentUserCommands.put(userId, command);

isFinished = command.execute(update, true);

} else if (currentUserCommands.containsKey(userId)) { // user continues some command

isFinished = currentUserCommands.get(userId).execute(update, false);

}

if (isFinished) {

currentUserCommands.remove(userId);

}

}

**Продолжение приложения А**

@Override

public String getBotUsername() {

return botUsername;

}

@Override

public String getBotToken() {

return botToken;

}

}

**Интерфейс BotServise.java**

package reserve.bot.bot;

import org.telegram.telegrambots.meta.api.methods.send.SendPhoto;

import org.telegram.telegrambots.meta.api.objects.Update;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboard;

import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import java.util.List;

public interface BotService {

void sendText(Long chatId, String text);

**Продолжение приложения А**

void sendWithKeyboard(Long chatId, String text, List<Button> buttons);

void sendMarkup(Long chatId, String text, ReplyKeyboard markup);

void sendPhoto(SendPhoto sendPhoto, List<Button> buttons);

String downloadPhoto(Update update) throws TelegramApiException;

}

**Класс BotServiceImpl.java**

package reserve.bot.bot;

import lombok.RequiredArgsConstructor;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

import org.telegram.telegrambots.meta.api.methods.GetFile;

import org.telegram.telegrambots.meta.api.methods.ParseMode;

import org.telegram.telegrambots.meta.api.methods.send.SendMessage;

import org.telegram.telegrambots.meta.api.methods.send.SendPhoto;

import org.telegram.telegrambots.meta.api.objects.Update;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.InlineKeyboardMarkup;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboard;

import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardButton;

import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

**Продолжение приложения А**

@RequiredArgsConstructor

@Component

public class BotServiceImpl implements BotService {

private final Bot bot;

@Value("${bot.images-dir}")

private String imagesDir;

@Override

public void sendText(Long chatId, String text) {

SendMessage send = new SendMessage();

send.setChatId(chatId.toString());

send.setText(text);

send.setParseMode(ParseMode.MARKDOWN);

try {

bot.execute(send);

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

@Override

public void sendWithKeyboard(Long chatId, String text, List<Button> buttons) {

SendMessage send = new SendMessage();

send.setChatId(chatId.toString());

send.setText(text);

**Продолжение приложения А**

send.setReplyMarkup(inlineKeyboard(buttons));

try {

bot.execute(send);

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

@Override

public void sendMarkup(Long chatId, String text, ReplyKeyboard markup) {

SendMessage send = new SendMessage();

send.setChatId(chatId.toString());

send.setText(text);

send.setParseMode(ParseMode.MARKDOWN);

send.setReplyMarkup(markup);

try {

bot.execute(send);

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

@Override

public void sendPhoto(SendPhoto sendPhoto, List<Button> buttons) {

sendPhoto.setReplyMarkup(inlineKeyboard(buttons));

sendPhoto.setParseMode(ParseMode.MARKDOWN);

**Продолжение приложения А**

try {

bot.execute(sendPhoto);

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

@Override

public String downloadPhoto(Update update) throws TelegramApiException {

String fileId = update.getMessage().getPhoto().get(2).getFileId();

String filePath = imagesDir + "/" + fileId + ".jpeg";

bot.downloadFile(bot.execute(new GetFile(fileId)), new File(filePath));

return filePath;

}

private InlineKeyboardMarkup inlineKeyboard(List<Button> buttons) {

InlineKeyboardMarkup inlineKeyboardMarkup = new InlineKeyboardMarkup();

List<List<InlineKeyboardButton>> totalList = new ArrayList<>();

for (Button button : buttons) {

List<InlineKeyboardButton> keyboardButtonRow = new ArrayList<>();

InlineKeyboardButton inlineButton = new InlineKeyboardButton(button.getLabel());

inlineButton.setCallbackData(button.getCallBack());

keyboardButtonRow.add(inlineButton);

totalList.add(keyboardButtonRow);

}

**Продолжение приложения А**

inlineKeyboardMarkup.setKeyboard(totalList);

return inlineKeyboardMarkup;

}

}

**Класс Button.java**

package reserve.bot.bot;

import lombok.Data;

@Data

public class Button {

String label;

String callBack;

public Button(String callBack, String label) {

this.callBack = callBack;

this.label = label;

}

}

**Класс Runner.java**

package reserve.bot.bot;

import lombok.RequiredArgsConstructor;

import org.springframework.boot.context.event.ApplicationReadyEvent;

**Продолжение приложения А**

import org.springframework.context.event.EventListener;

import org.springframework.stereotype.Component;

import org.telegram.telegrambots.meta.TelegramBotsApi;

import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;

@RequiredArgsConstructor

@Component

public class Runner {

private final Bot bot;

@EventListener(ApplicationReadyEvent.class)

public void registerBot() {

try {

TelegramBotsApi botsApi = new TelegramBotsApi(DefaultBotSession.class);

botsApi.registerBot(bot);

} catch (TelegramApiException e) {

e.printStackTrace();

}

}

}

**Окончание приложения А**

**Класс UpdateUtill.java**

package reserve.bot.bot;

import org.telegram.telegrambots.meta.api.objects.Update;

public class UpdateUtil {

public static long getUserId(Update update) {

return update.hasCallbackQuery()

? update.getCallbackQuery().getMessage().getChatId()

: update.getMessage().getChatId();

}

public static String getInput(Update update) {

return update.hasCallbackQuery()

? update.getCallbackQuery().getData()

: update.getMessage().getText();

}

}