

# Требования к проекту Quiz Bot

Шихалев Алексей, Емешкин Максим

## Оглавление

<b>Введение.....</b>	<b>3</b>
<b>1. Общие требования к разработке.....</b>	<b>4</b>
1.1. Контроль версий и совместная работа.....	4
1.2. Целевые артефакты.....	4
<b>2. Пользовательские требования.....</b>	<b>5</b>
2.1. Авторизация.....	5
2.2. Управление вопросами.....	5
<b>3. Функциональные требования.....</b>	<b>6</b>
3.1. Авторизация и роли.....	6
3.2. Доступ к защищенным эндпоинтам для Admin.....	6
3.3. Список защищенных эндпоинтов.....	7
3.4. Health check.....	7
<b>4. Описание команд для взаимодействия пользователя с Ботом.....</b>	<b>8</b>
4.1 Авторизация.....	8
4.2 Список команд (меню).....	8
4.3 Добавление нового вопроса.....	9
4.4. Просмотр списка вопросов по тегу.....	10
4.5 Удаление вопроса.....	10
4.6 Добавление тега.....	11
4.7 Удаление тега.....	11
4.8 Список тегов.....	12
4.9 Отправка случайного вопроса по расписанию.....	12
4.10 Отмена расписания.....	13
4.11 Запрос случайного вопроса.....	13
4.12 Запрос случайного вопроса по выбранной теме (тегу).....	14
4.13 Статистика пользователя.....	14
4.14 Отображение счёта по теме (тегу).....	14
4.15 Сброс счета.....	15
<b>5. Нефункциональные требования.....</b>	<b>16</b>
5.1. Производительность.....	16

5.2. Ограничения.....	16
<b>6. Технологические требования.....</b>	<b>17</b>
<b>7. Руководство по сдаче проекта.....</b>	<b>18</b>
7.1. Требования.....	18
7.2. Архитектура.....	18
7.3. Полный проект.....	18
<b>8. Глоссарий.....</b>	<b>19</b>
8.1. Основные понятия.....	19
8.2. Идентификаторы и данные.....	19
8.3. Технические термины.....	19
8.5. Документация и инфраструктура.....	20
8.6. Требования.....	20

# Введение

Целью проекта является разработка Telegram-бота для проведения викторин, который позволяет пользователям создавать вопросы, получать случайный вопрос по запросу, настраивать автоматическую отправку одного случайного вопроса по расписанию, просматривать и сбрасывать счет.

# 1. Общие требования к разработке

## 1.1. Контроль версий и совместная работа

1. Весь код должен быть загружен в единый репозиторий GitHub.
2. Репозиторий должен содержать файл README.md с подробным руководством по:
  - Настройке проекта локально.
  - Сборке и запуску приложения.
  - Развертыванию приложения с помощью Docker.
  - Взаимодействию с Telegram-ботом.
3. История коммитов должна отражать значимый вклад всех участников команды.

## 1.2. Целевые артефакты

- Приложение должно собираться в исполняемый fat JAR как основной артефакт.
- Приложение должно быть развертываемым в виде Docker-контейнера.
- Для управления зависимостями, сборкой и упаковкой приложения необходимо использовать систему сборки Maven.

## 2. Пользовательские требования

### 2.1. Авторизация

Пользователь должен иметь возможность начать работу с Ботом через отправку команды \start в чате с Ботом.

### 2.2. Управление вопросами

Пользователь должен иметь возможность:

- добавлять и удалять вопросы с ответами и тегами;
- получить случайный вопрос с вариантами ответов по команде;
- получить случайный вопрос с вариантами ответов по определенной теме (тегу);
- установить себе получение случайного вопроса по расписанию;
- посмотреть текущий общий счёт и счёт по определенной теме;

Администратор должен обладать всеми возможностями обычного пользователя, а также иметь возможность:

- назначать/отзывать права администратора;
- сбрасывать счёт любого пользователя;
- просматривать список пользователей.

## 3. Функциональные требования

### 3.1. Авторизация и роли

1. Система должна поддерживать два типа учетных записей:
  - **User** - роль для Пользователя. Позволяет иметь доступ к основным функциям Бота (см. п. 4.1 - 4.15).
  - **Admin** - роль для Администратора. Позволяет иметь доступ ко всем функциям Системы: просмотр списка всех пользователей и управление правами (назначение роли Admin для User, снятие с пользователя роли Admin).
2. Механизмы аутентификации  
Авторизация через Telegram:
  - При вызове команды /start из Telegram-чата с Ботом Система получает user\_id Пользователя от Telegram API.
  - Полученный user\_id система сохраняет в БД с ролью по умолчанию User.
  - Ответ бота:  
[Добро пожаловать, <user\_id>!  
Используйте /help для списка команд. ]
  - После авторизации Пользователь получает возможность взаимодействовать с Ботом.

### 3.2. Доступ к защищенным эндпоинтам для Admin

- Система должна обеспечивать ограниченный доступ к административным REST-эндпоинтам по протоколу HTTP с использованием базовой аутентификации (Basic Authentication).
- Запросы без заголовка Authorization: Basic <credentials> или с некорректными учётными данными должны возвращать 401 Unauthorized.
- Для доступа требуется роль Admin. Пользователи с ролью User получают 403 Forbidden.
- Учетные данные (логины и пароли) хранятся в БД в зашифрованном виде.

### 3.3. Список защищенных эндпоинтов

- GET /admin/users — получение списка всех пользователей.

- POST /admin/users/{userId}/promote — назначение роли Admin пользователю с user\_id.
- POST /admin/users/{userId}/demote — снятие роли Admin (неприменимо к текущему администратору).
- POST /admin/users/{userId}/reset\_score – сброс счета пользователя.

### 3.4. Health check

Добавить команду /healthcheck, которая:

1. Возвращает статус приложения.
2. Выводит список авторов-студентов (ФИО, группа).

## 4. Описание команд для взаимодействия пользователя с Ботом

### 4.1 Авторизация

Команда: `/start`

Описание: Запускает бота и авторизует пользователя.

Реакция Бота:

1. Получает `user_id` от Telegram API.
2. Проверяет наличие пользователя в БД:
  - 2.1. Если нет — создает нового пользователя с ролью *User*.
3. Приветствует пользователя:

{ 🎉 Добро пожаловать, {user\_id}! 🎉 }

Этот бот поможет тебе создавать и проходить викторины!

Используй `/help`, чтобы увидеть список команд. }

### 4.2 Список команд (меню)

Команда: `/help`

Описание: Показывает список доступных команд.

Реакция Бота:

{ 📋 Список команд:

`/add_question` — создать новый вопрос с вариантами ответов;

`/show_questions_by_tag <тег>` — показать список всех вопросов,

созданных пользователем, связанных с указанным тегом.

`/delete_question <ID>` — удалить вопрос;

`/add_tag` — добавить новый тег;

`/delete_tag <тег>` — удалить существующий тег, созданный пользователем;

`/list_tags` — список всех тегов;



`/schedule` — настройка автоматической отправки случайных вопросов в чат по расписанию;

`/unschedule` — отменить текущее расписание;

`/random` — случайный вопрос;

`/random_by_tag <тег>` — случайный вопрос по выбранной теме;

`/score` — вывод общего количества баллов;

`/score_by_tag <тег>` — вывод количества баллов по выбранной теме;

`/reset_score` — обнулить счет.

}

### 4.3 Добавление нового вопроса

Команда: `/add_question`

Описание: Создает новый вопрос с вариантами ответов и тегами.

Реакция бота:






1. Запрашивает текст вопроса: {📝 Введите текст вопроса (макс. 200 символов):}
  - 1.1. Некорректный ввод (пусто/длиннее 200 символов): {❌ Текст вопроса должен содержать от 1 до 200 символов.}
2. Запрашивает варианты ответов (от 2 до 4): {📋 Введите вариант 1:  
(После ввода всех вариантов): ✅ Добавить еще вариант? (Да/Нет)}
3. Указывает правильный ответ: {✅ Введите номер правильного варианта (1-{N}):}
4. Добавляет теги: {🏷 Введите теги (списком через запятую, например: \_история, наука):}
5. Подтверждение: {✅ Вопрос сохранен! 🆔 ID: Q123 Теги: история, наука}

## 4.4. Просмотр списка вопросов по тегу

Команда: `/show_questions_by_tag <тег>`

Описание: Показывает список всех вопросов, созданных пользователем, связанных с указанным тегом.

Реакция бота:






1. Проверяет наличие тега в системе:
  - Если тег не найден:  
 Тег «{тег}» не существует. Используйте `/list_tags` для просмотра доступных тегов.
  - Если тег найден:  
 Список вопросов по тегу «{тег}» (всего {N}):  
Выводит вопросы в формате:
    -  ID: Q123 — {Текст вопроса (первые 50 символов)...}
    -  ID: Q456 — {Текст вопроса...}
  - Если вопросов нет:  
 По тегу «{тег}» пока нет ваших вопросов.

## 4.5 Удаление вопроса

Команда: `/delete_question <ID>`

Описание: Удаляет вопрос, созданный пользователем по его ID.

Реакция бота:

1. Проверяет наличие вопроса:
  - 1.1. Если не найден: {  Вопрос с ID <ID> не существует. }
  - 1.2. Если найден, но принадлежит другому пользователю: {  Вопрос с ID <ID> создан другим пользователем. }
  - 1.3. Если найден → подтверждение: {  Удалить вопрос: «{Текст вопроса}»? (Да/Нет) }
    - 1.3.1. Да: {  Вопрос удален. }
    - 1.3.2. Нет: {  Отменено. }

## 4.6 Добавление тега

Команда: `/add_tag`

Описание: Создает новый тег для категоризации вопросов.

Реакция бота:

1. Запрашивает название тега: { 🗒 Введите название тега (англ./рус., без пробелов): }
  - 1.1. Некорректный ввод (тег уже существует или содержит пробелы): { ❌ Тег «{тег}» уже есть или содержит пробелы. }
2. Подтверждение: { ✅ Тег «{тег}» добавлен! }

## 4.7 Удаление тега

Команда: `/delete_tag <тег>`

Описание: Удаляет тег из системы, созданный пользователем.

Реакция бота:

1. Проверяет существование тега:
  - 1.1. Если тег не найден: { ❌ Тег «{тег}» не существует. }
  - 1.2. Если найден, но создан другим пользователем: {  
❌ Тег «{тег}» создан другим пользователем. }
  - 1.3. Если найден → предупреждение: { ! Удаление тега «{тег}» также удалит все вопросы. Продолжить? (Да/Нет) }
    - 1.3.1. Да: { ✅ Тег «{тег}» удален. }
    - 1.3.2. Нет: { ! Отменено. }

## 4.8 Список тегов

Команда: `/list_tags`

Описание: Показывает все доступные теги.

Реакция бота:

{🔖 Доступные теги (всего {N}):}

- история (используется в 15 вопросах)
- наука (10)
- спорт (5)}


## 4.9 Отправка случайного вопроса по расписанию




Команда: `/schedule`

Описание: Настраивает автоматическую отправку случайных вопросов в чат по расписанию.

Реакция Бота:

1. Запрашивает параметры: {🕒 Укажите время первого вопроса (например, 14:00):}
  - 1.1. Корректный ввод: ЧЧ:ММ (24-часовой формат).
  - 1.2. Некорректный ввод: {❌ Неверный формат времени.  
Используйте ЧЧ:ММ (например, 14:00).}
2. Запрашивает периодичность: {🔄 Выберите периодичность: 1. Ежедневно 2. Еженедельно 3. Каждые N часов (введите число)}

2.1. Пример для «Каждые N часов»: {  Введите интервал в часах (1-24): }





3. Подтверждение: {  Расписание настроено!  Первый вопрос — завтра в 14:00.  Повтор: ежедневно. }

## 4.10 Отмена расписания

Команда: `/unschedule`

Описание: Отключает автоматическую отправку вопросов.

Реакция бота:



1. Если расписание не активно: {  Автоотправка вопросов не настроена. }
2. Если активно → подтверждение: {  Отменить текущее расписание (следующий вопрос — завтра в 14:00)? (Да/Нет) }
  - 2.1. Да: {  Расписание отменено. }
  - 2.2. Нет: {  Отменено. }

## 4.11 Запрос случайного вопроса

Команда: `/random`

Описание: Задает пользователю случайный вопрос из базы.

Реакция Бота:

1. Выбирает случайный вопрос и отправляет: {  Случайный вопрос: {Текст вопроса} 1. Вариант 1 2. Вариант 2 3. Вариант 3 }
2. Ожидает ответа (таймер 30 сек):
  - 2.1. При правильном ответе: {  Верно! +10 баллов. }

2.2. При неверном: {❌ Неверно. Правильный ответ: {Вариант}.}

## 4.12 Запрос случайного вопроса по выбранной теме (тегу)

Команда: `/random_by_tag <тег>`

Описание: Отправляет вопрос по выбранной теме.

Реакция Бота:

1. Проверяет наличие тега:

1.1. Если тега нет → сообщение: {❌ Тег «{тег}» не найден.

Доступные теги: `/list_tags.`}

2. Отправляет вопрос: {📄 Вопрос по теме «{тег}»: {Текст вопроса} 1.

Вариант 1 2. Вариант 2 3. Вариант 3}

3. Ожидает ответа (таймер 30 сек):

3.1. При правильном ответе: {✅ Верно! +10 баллов.}

3.2. При неверном: {❌ Неверно. Правильный ответ: {Вариант}.}

## 4.13 Статистика пользователя

Команда: `/score`

Описание: Показывает общее количество баллов.

Реакция Бота: {📊 Ваш счёт: 🏆 Всего баллов: 250 }

## 4.14 Отображение счёта по теме (тегу)

Команда: `/score_by_tag <тег>`

Описание: Показывает статистику по выбранной теме.




Реакция Бота: {  Баллы по теме <тег>: 80/100 }

#### 4.15 Сброс счета

Команда: `/reset_score`

Описание: Обнуляет все баллы пользователя.

Реакция бота:

1. Запрашивает подтверждение: {  Вы уверены, что хотите сбросить все баллы (текущий счет: 150)? (Да/Нет)}
- 1.1. Да: {  Счет обнулен. }
- 1.2. Нет: {  Отменено. }

## 5. Нефункциональные требования

### 5.1. Производительность

- Время отклика Бота на команды пользователя не должно превышать 600 мс.
- Время обработки HTTP-запросов (для роли Admin) не должно превышать 1 секунды.
- Напоминания должны доставляться с задержкой не более 5 секунд от запланированного времени.
- Система должна поддерживать до 100000 одновременных пользователей.

### 5.2. Ограничения

5.3.1. Максимальное количество вопросов по любым темам, лежащих в базе данных - 1000.

5.3.2. Максимальное количество символов для вопроса и для ответа - 200.

5.3.3. Количество ответов у каждого вопроса фиксированное и равно 4.

5.3.4. Максимальное количество тегов у одного вопроса – 5.

5.3.5. Максимальное количество тегов в базе данных – 500.



## 6. Технологические требования

- **Язык программирования:** Java 23.
- **Фреймворк:** Spring 6.2.3
  - Spring REST.
  - JPA(+Hibernate).
  - Spring Rest Docs.
- **Telegram API:** Приложение должно интегрироваться с Telegram API для предоставления функциональности бота.
- **Брокер сообщений:** Kafka.
- **База данных:** PostgreSQL.
- **Контейнеризация:** Docker, Docker Compose.

## 7. Руководство по сдаче проекта

### 7.1. Требования

Предоставить документ с требованиями, который включает:

- цель и функциональность приложения,
- ключевые функции Telegram-Бота,
- нефункциональные требования,
- любые предположения или ограничения.

### 7.2. Архитектура

Предоставить документ по архитектуре, который включает:

- высокоуровневый обзор дизайна системы,
- диаграммы (например, компонентные диаграммы, диаграммы последовательностей или блок-схемы) для иллюстрации архитектуры,
- обоснование выбранного стека,
- описание того, как приложение будет строиться, развертываться и запускаться.

### 7.3. Полный проект

Предоставить полный проект в виде:

- ссылки на репозиторий GitHub с полной кодовой базой,
- ссылки на Docker Hub, где опубликован Docker-образ,
- ссылки на имя пользователя Telegram-Бота для тестирования и взаимодействия.

## 8. Глоссарий

### 8.1. Основные понятия

**Система** – Программный продукт, включающий Telegram-бота и серверную часть для управления задачами.

**Бот** – Telegram-бот для создания, редактирования и управления задачами.

**User** – Роль, позволяющая работать со своими задачами (создание, просмотр, редактирование).

**Admin** – Роль с расширенными правами (управление пользователями, доступ к защищенным HTTP-эндпоинтам).

### 8.2. Идентификаторы и данные

**user\_id** – Уникальный идентификатор пользователя в Telegram, используемый для авторизации.

**БД (База данных)** – Хранилище информации о пользователях, задачах и настройках.

**Идентификатор, ID** (англ. data name, identifier — опознаватель) — уникальный признак объекта, позволяющий отличать его от других объектов, то есть идентифицировать.

**Логин** – идентификатор пользователя (учётной записи) в компьютерных системах.

**Роль** – комплект прав доступа, необходимых для выполнения конкретных функций, который предоставляется пользователю Системы.

**Система управления базами данных, сокр. СУБД** (англ. Database Management System, сокр. DBMS) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием БД.

### 8.3. Технические термины

**Docker** – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

**Docker-контейнер** – Изолированная среда для запуска приложения.

**HTTP** (англ. Hypertext Transfer Protocol – «протокол передачи гипертекста») – протокол уровня приложений для распределённых,

объединённых, гипермедийных информационных систем, используемый в глобальной информационной инициативе Всемирной паутины

**Jakarta Persistence API (JPA;** ранее Java Persistence API) – спецификация API Jakarta EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных.

**JSON** (англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.

**Maven** – Инструмент для сборки проекта и управления зависимостями.

**Telegram API** – Интерфейс для взаимодействия с Telegram.

**Spring** – универсальный фреймворк с открытым исходным кодом для Java-платформы.

**Spring REST** – это модуль фреймворка Spring, предназначенный для создания RESTful (Representational State Transfer) веб-сервисов.

**Артефакт** – любой результат работы разработчика, который создается в процессе сборки, развертывания или выполнения программы.

**Эндпоинт (Endpoint)** – URL для взаимодействия с сервером.

## 8.5. Документация и инфраструктура

**Docker Hub** – Платформа для хранения Docker-образов.

**GitHub-репозиторий** – Хранилище кода с историей изменений.

**README.md** – Файл с инструкциями по настройке и запуску проекта.

## 8.6. Требования

**Нефункциональные требования** – Требования к производительности, безопасности, ограничениям.

**Команда (бота)** – Инструкция, которую пользователь отправляет боту.

**Функциональные требования** – Требования к возможностям Системы.