МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ О ПРОХОЖДЕНИИ

УЧЕБНОЙ ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

на базе Высшей школы компьютерных наук и искусственного интеллекта

Выполнил ФИО Питерцев Аксен Михайлович

студент очной формы обучения 1 курса  
направления подготовки 02.03.03 «Математическое обеспечение  
и администрирование информационных систем»  
профиль обучения «Разработка баз данных  
и интернет-приложений»

Руководитель практики

Ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Полковский О.А

г. Калининград  
2025 г.

Оглавление

[ВВЕДЕНИЕ 3](#_ovlz1518vapf)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_3e3k0m2z7hm5)

Telegram-бот [4](#_ehqcvujnlsnu)

[ГЛАВА 2. ЗАДАНИЕ НА ПРАКТИКУ 5](#_ldkmkiwkup9f)

[Задача «Создание telegram - бота для кофейни»](#_nzvu0kdrb9nc) [5](#_wi0xd0c0dh0p)

[Глава 3. Выполнение задания на практику](#_ahgru5clbc0c) 6

Использованные технологии 6

Структура базы данных 6

Описание реализованных команд 7

Команды, реализованные напарником 9

Реферальная система 10

Сложности и их решения 10

[ЗАКЛЮЧЕНИЕ 1](#_3lrj8oncba7v)1

[СПИСОК ЛИТЕРАТУРЫ 1](#_uskxn07a90iv)2

[ПРИЛОЖЕНИЕ 1 1](#_onb4rddw0r5w)3

**ВВЕДЕНИЕ**

Вид практики – Учебная технологическая (проектно-технологическая) практика (далее Учебная практика).

Цель учебной практики: получение первичных профессиональных умений навыков.

Задачи учебной практики:

1. Закрепление и углубление теоретических знаний в области информационных технологий;
2. Приобретение и развитие первичных профессиональных навыков и умений в области прикладной математики и информатики.

**ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Telegram-бот — это специальная программа, взаимодействующая с пользователем через интерфейс приложения Telegram. Для его реализации используется Telegram Bot API, предоставляющий удобные методы отправки сообщений, обработки команд и работы с пользователями.  
  
 Для хранения данных был использован SQLite — встроенная реляционная база данных, не требующая отдельного сервера. Она подходит для малых и средних проектов, легко интегрируется с Python и обеспечивает хранение данных в одном файле.

**ГЛАВА 2. ЗАДАНИЕ НА ПРАКТИКУ**

В рамках учебной практики требовалось реализовать Telegram-бота, соответствующего следующим условиям:

1. Гость может оставить анонимный отзыв, который будет виден только администратору.
2. В боте реализована программа лояльности: при покупке любого напитка пользователь получает +1 балл. Набрав 5 баллов, гость получает бесплатный напиток. После использования бесплатного напитка с его счёта списывается 5 баллов.
3. Бот в приветственном сообщении отправляет ссылки на социальные сети.
4. Гость может сделать заказ через бота (бариста получит сообщение с заказом в свободной форме). (без оплаты)
5. Реферальная программа: гости могут делиться ссылками на бота со своими друзьями и получать дополнительные баллы за каждого нового клиента, который совершит покупку. Отправитель получает 2 балла, а новый клиент - 3 балла.

**Глава 3. Выполнение задания на практику**

**Использованные технологии**

- Язык программирования: Python  
- Telegram API: библиотека python-telegram-bot  
- База данных: SQLite3  
- Среда разработки: Visual Studio Code  
- Система контроля версий: GitHub

**Структура базы данных**

Проект использует базу данных, состоящую из следующих таблиц:

**Menu:**

| **id** | **available** | **type** | **name** | **price** | **option** | **deleted** |
| --- | --- | --- | --- | --- | --- | --- |
| integer, autoincrement, primary key | integer | string | string | integer | string | integer |

**Orders:**

| **id** | **date** | **active** | **code** | **drink** |
| --- | --- | --- | --- | --- |
| integer | string | integer | integer | integer |

**Staff:**

| **id** | **username** | **is\_working** | **fired** |
| --- | --- | --- | --- |
| integer, autoincrement, primary key | string | integer | integer |

**Users:**

| id | points | linked\_by |
| --- | --- | --- |
| integer, autoincrement, primary key | integer | string |

**Описание реализованных команд**

**/start**

Отправляет приветственное сообщение с ссылками на соцсети и регистрирует пользователя в базе данных. Если пользователь перешёл по реферальной ссылке, его никнейм из неё сохраняется в поле linked\_by.

**/feedback**

Позволяет пользователю отправить анонимный отзыв. Сообщение пересылается администратору без указания отправителя(с помощью функций forward\_message), в логах также не отображается информации о пользователе, в базе данных ничего не сохраняется.

**/invite**

Генерирует и отправляет ссылку вида http://t.me/logos\_coffee\_bot?start=user\_id, где user\_id - никнейм пользователя, запрашивающего ссылку. User\_id будет вписан в поле linked\_by пользователя, который воспользуется реферальной ссылкой.



Рисунок 1. Функция invite

**/order**

Позволяет пользователю оформить заказ, посмотреть корзину, в которой можно удалить или добавить товар. Реализовано с помощью ConversationHandler и InlineKeyboard. Заказ сохраняется в query.data пользователя. В конце заказ отправляется бариста, который находится в рабочем статусе, в виде всех позиций заказа с ценой, итоговой суммой и уникальным кодом заказа.

**/delete\_barista**

Удаляет бариста(устанавливает значение 1 в поле fired), работает для администраторов. После команды указывается user\_id сотрудника.

**/add\_barista**

Добавляет нового бариста, работает для администраторов. После команды указывается user\_id и никнейм пользователя, которого нужно назначить сотрудником.

**/change\_status**

Меняет рабочий статус бариста (is\_working) на противоположный. В рабочем статусе заказы приходят, в нерабочем — нет.

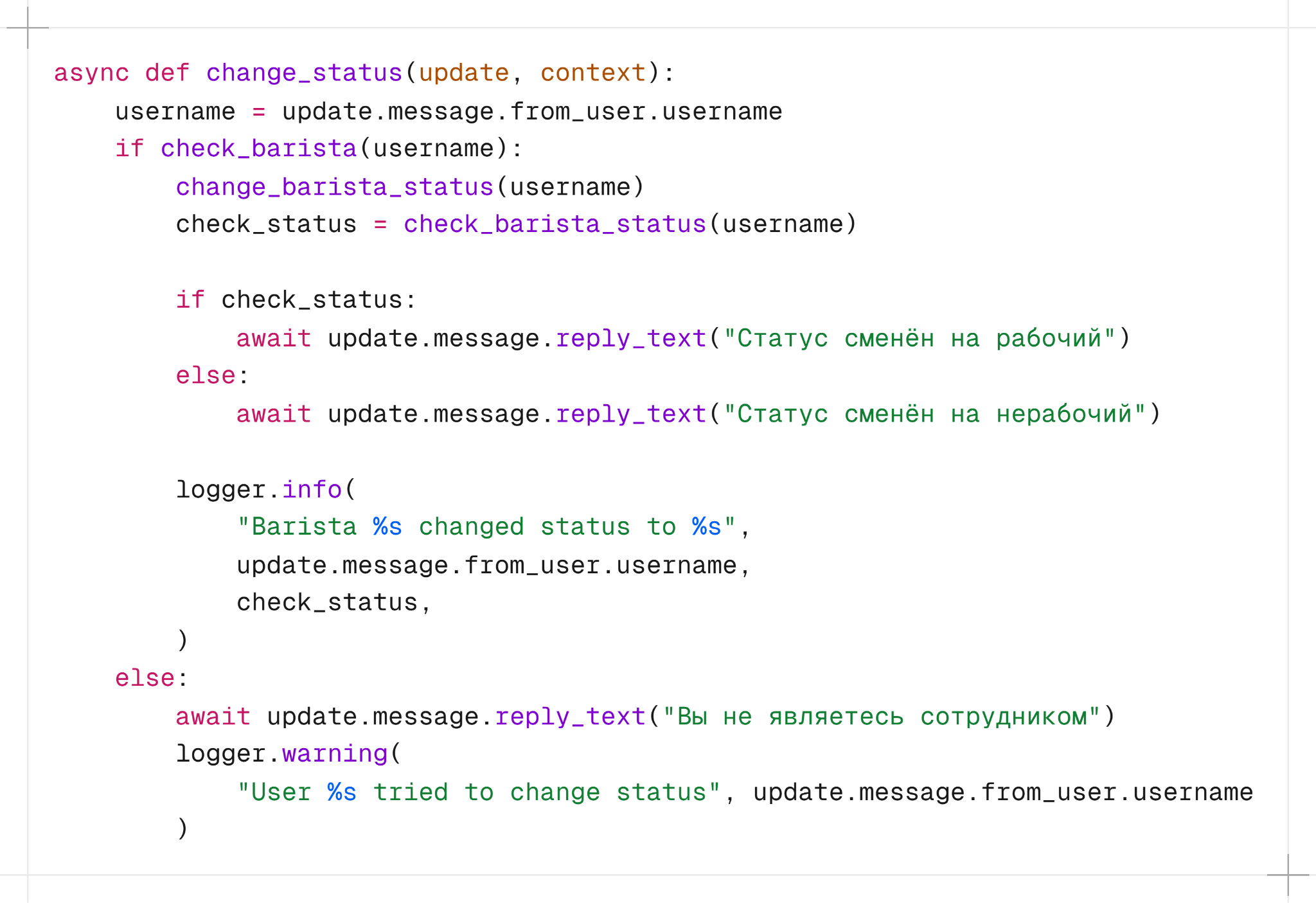


Рисунок 2. Функция change\_status

**Команды, реализованные напарником**

/menu

Отображает список доступных товаров.

/order

Совместно вели работу

/add\_product

Добавляет выбранный товар в корзину.

/delete\_product

Удаляет указанный товар из корзины.

/change\_availability

Изменяет доступность позиции меню. Команда доступна администраторам.

/points

Показывает количество накопленных бонусов пользователя.

/cancel

Завершает сессию пользователя (например, очищает корзину).

**Реферальная система**

Реализована система лояльности:  
- За покупку напитка начисляется 1 балл  
- За приглашение друга начисляется 2 балла при его первой покупке. При этом из поля linked\_by он отвязывается и поле блокируется.  
- Новому пользователю начисляется 3 балла при регистрации по реферальной ссылке

**Сложности и их решения**

Основная сложность возникла при реализации команды /order и механизма постраничного отображения меню. Были протестированы несколько вариантов, прежде чем удалось реализовать корректную навигацию по меню с помощью кнопок «далее» и «назад», а конкретно, использовался срез списка. Списком выступает доступное меню товаров(товары, где deleted == 0, available == 1). При нажатии кнопок «далее»(«назад») индексы среза сдвигаются на необходимое количество.

Также полный код главного файла [main.py](http://main.py) указан в Приложении 1

**ЗАКЛЮЧЕНИЕ**

В ходе учебной технологической практики были закреплены знания в области разработки интернет-приложений, освоены основы взаимодействия с Telegram Bot API, проектирования базы данных и клиент-серверного взаимодействия.   
 В результате практики были усовершенствованы мои компетенции, закреплены теоретические навыки.

В ходе учебной практики я научился: выполнять поставленную задачу, следуя условию, изменять готовое решения по мере нахождения ошибок, искать альтернативные пути решения задач.  
 Работа в команде позволила эффективно разделить обязанности, что способствовало успешному выполнению поставленных задач. Полученные знания и навыки могут быть применены в дальнейших учебных и проектных работах.

# 

# СПИСОК ЛИТЕРАТУРЫ

**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

1. Python-telegram-bot documentation: сайт. – URL: https://docs.python-telegram-bot.org/en/v21.6/index.html (дата обращения: 07.07.2025). – Текст: электронный.
2. Telegram Bot API: сайт. – URL: https://core.telegram.org/bots/api (дата обращения: 07.07.2025). – Текст: электронный.
3. SQLite Documentation: сайт. - URL: https://www.sqlite.org/docs.html (дата обращения: 07.07.2025). - Текст: электронный
4. Python Documentation: сайт. - URL: https://docs.python.org/3.11/ (дата обращения: 07.07.2025). - Текст: электронный

## ПРИЛОЖЕНИЕ 1

| 0 | **from** imports **import** \*  1 | **import** config  2 | ​  3 | **from** telegram.ext **import** (  4 | CallbackQueryHandler,  5 | ConversationHandler,  6 | ApplicationBuilder,  7 | CommandHandler,  8 | MessageHandler,  9 | filters,  10| )  11| ​  12| **from** warnings **import** filterwarnings  13| **from** telegram.warnings **import** PTBUserWarning  14| ​  15| filterwarnings(  16| action="ignore", message=r".\*CallbackQueryHandler", category=PTBUserWarning  17| )  18| ​  19| **if** \_\_name\_\_ == "\_\_main\_\_":  20| application = ApplicationBuilder().token(config.TOKEN).build()  21| ​  22| # user commands  23| application.add\_handler(CommandHandler("start", start))  24| application.add\_handler(CommandHandler("invite", invite))  25| application.add\_handler(CommandHandler("menu", menu))  26| application.add\_handler(CommandHandler("points", points))  27| application.add\_handler(CommandHandler("get\_id", get\_id))  28| application.add\_handler(CommandHandler("help", help))  29| ​  30| feedback\_handler = ConversationHandler(  31| entry\_points=[  32| CommandHandler("feedback", feedback),  33| ],  34| states={  35| FORWARD\_FEEDBACK: [  36| MessageHandler(filters.TEXT & ~filters.COMMAND, forward\_feedback)  37| ],  38| },  39| fallbacks=[MessageHandler(filters.COMMAND, cancel)],  40| )  41| application.add\_handler(feedback\_handler)  42| ​  43| order\_handler = ConversationHandler(  44| entry\_points=[CommandHandler("order", order\_product\_type)],  45| states={  46| ORDER\_PRODUCT\_NAME: [CallbackQueryHandler(order\_product\_name)],  47| ORDER\_PRODUCT\_OPTION: [CallbackQueryHandler(order\_product\_option)],  48| ORDER\_PRODUCT: [CallbackQueryHandler(order\_product)],  49| CART\_CHANGE: [CallbackQueryHandler(cart\_change)],  50| CART\_DELETE: [CallbackQueryHandler(cart\_delete)],  51| },  52| fallbacks=[MessageHandler(filters.COMMAND, cancel)],  53| )  54| application.add\_handler(order\_handler)  55| ​  56| # admin commands  57| application.add\_handler(CommandHandler("delete\_barista", delete\_barista))  58| application.add\_handler(CommandHandler("add\_barista", add\_barista))  59| ​  60| # barista commands  61| application.add\_handler(CommandHandler("change\_status", change\_status))  62| application.add\_handler(CommandHandler("ready", ready))  63| application.add\_handler(CommandHandler("done", done))  64| ​  65| add\_product\_handler = ConversationHandler(  66| entry\_points=[CommandHandler("add\_product", add\_product)],  67| states={  68| PRODUCT\_TYPE: [  69| CallbackQueryHandler(product\_type\_button),  70| MessageHandler(filters.TEXT & ~filters.COMMAND, product\_type),  71| ],  72| PRODUCT\_NAME: [  73| MessageHandler(filters.TEXT & ~filters.COMMAND, product\_name)  74| ],  75| PRODUCT\_OPTIONS: [  76| MessageHandler(filters.TEXT & ~filters.COMMAND, product\_options)  77| ],  78| PRODUCT\_PRICE: [  79| MessageHandler(filters.TEXT & ~filters.COMMAND, product\_price)  80| ],  81| },  82| fallbacks=[MessageHandler(filters.COMMAND, cancel)],  83| )  84| application.add\_handler(add\_product\_handler)  85| ​  86| delete\_product\_handler = ConversationHandler(  87| entry\_points=[CommandHandler("delete\_product", delete\_product)],  88| states={  89| DELETE\_PRODUCT\_NAME: [  90| MessageHandler(filters.TEXT & ~filters.COMMAND, delete\_product\_name)  91| ],  92| DELETE\_PRODUCT\_OPTIONS: [  93| MessageHandler(filters.TEXT & ~filters.COMMAND, delete\_product\_options)  94| ],  95| },  96| fallbacks=[MessageHandler(filters.COMMAND, cancel)],  97| )  98| application.add\_handler(delete\_product\_handler)  99| ​  100| change\_availability\_handler = ConversationHandler(  101| entry\_points=[  102| CommandHandler("change\_availability", change\_product\_availability)  103| ],  104| states={  105| AVAILABILITY\_PRODUCT\_NAME: [  106| MessageHandler(  107| filters.TEXT & ~filters.COMMAND, change\_product\_availability\_name  108| )  109| ],  110| AVAILABILITY\_PRODUCT\_OPTIONS: [  111| MessageHandler(  112| filters.TEXT & ~filters.COMMAND, change\_product\_availability\_options  113| )  114| ],  115| },  116| fallbacks=[MessageHandler(filters.COMMAND, cancel)],  117| )  118| application.add\_handler(change\_availability\_handler)  119| ​  120| application.run\_polling()  121| ​ |
| --- |