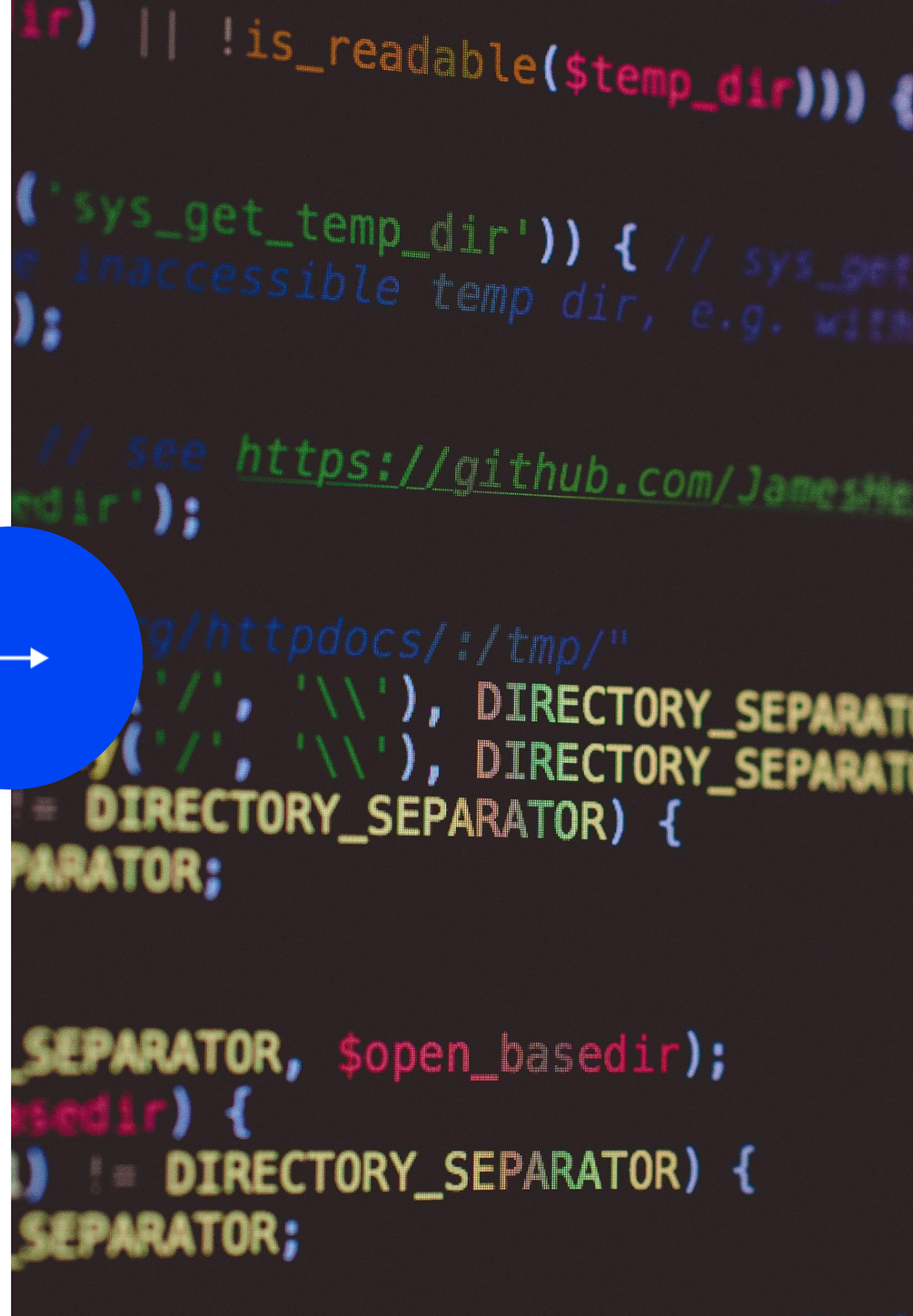
ITentika

Интеграция чат-бота в существующую архитектуру

Артур Чеканов, главный архитектор ITentika



Пара слов обо мне

Артур Чеканов

- Главный архитектор ITentika
- 14+ лет в IT
- Python в сердечке
- Сфера интересов: Solution Architecture, High Load, Data

О чем поговорим?

01 Как развивались подходы к взаимодействию с пользователем

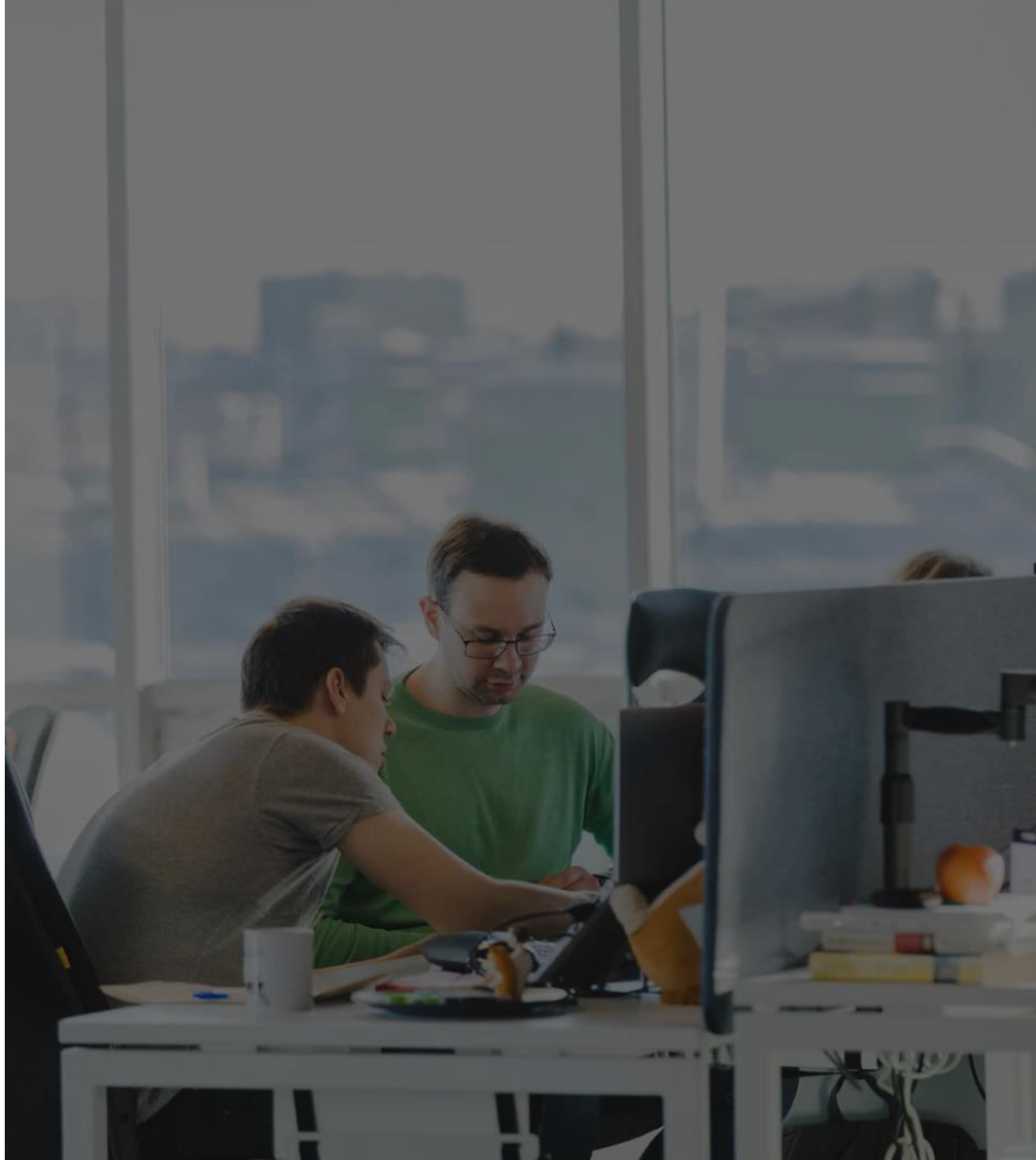
02 История одного продукта: как мы пришли к идее чат-бота

03 Как мы внедряли чат-бота в инфрастуктуру компании

04 Выводы

01

Как развивались подходы к взаимодействию с пользователем



Эволюция взаимодействия с пользователем

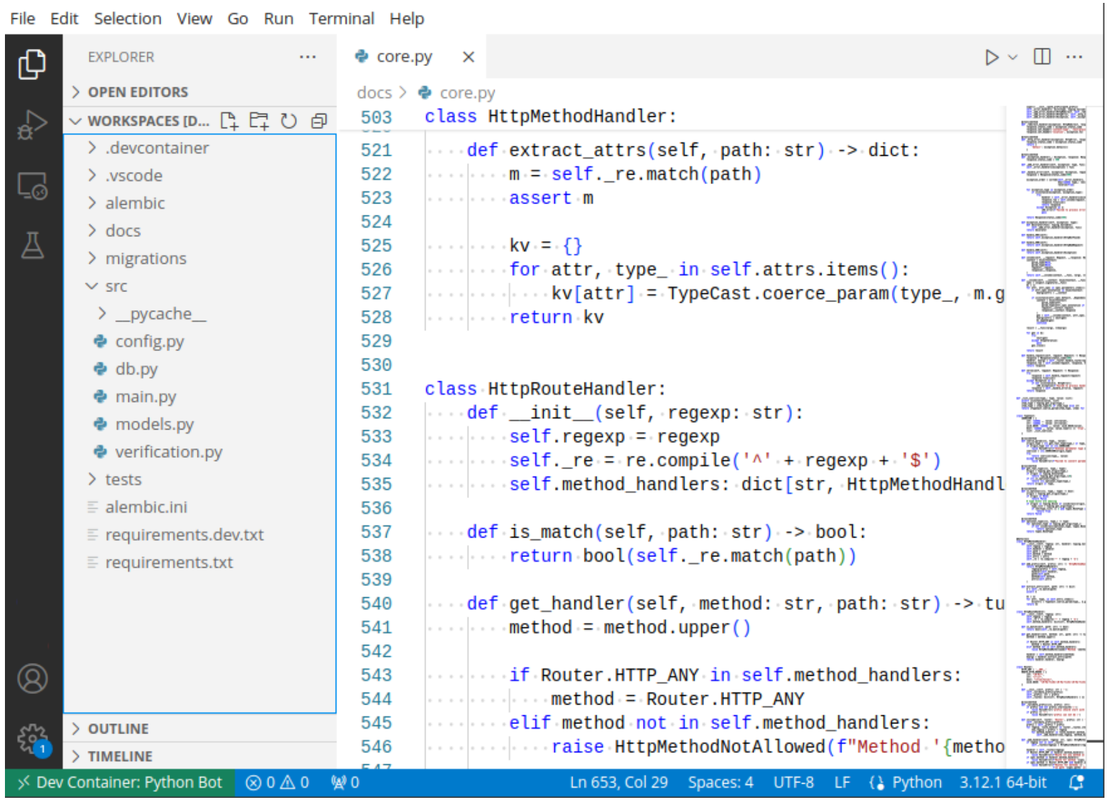
- Любой программный продукт в конечном итоге призван взаимодействовать с пользователем
- Приложения развивались с учетом повышения удобства использования
- Консольные приложения – самая простая и неудобная форма взаимодействия
- Десктопные приложения – кликаем мышкой
- Веб приложения – не нужно ничего устанавливать и настраивать
- Мобильные приложения – всегда под рукой
- PWA – всегда под рукой, но проще для разработчика
- Чат-боты – а почему бы и нет

```
[art@localhost ~]$ wget https://www.python.org/ftp/python/3.12.6/Python-3.12.6.t
ar.xz
--2024-09-24 16:17:35--  https://www.python.org/ftp/python/3.12.6/Python-3.12.6.
tar.xz
Распознаётся www.python.org (www.python.org)… 151.101.36.223, 2a04:4e42:9::223
Подключение к www.python.org (www.python.org)|151.101.36.223|:443... соединение
установлено.
HTTP-запрос отправлен. Ожидание ответа… 200 OK
Длина: 20434028 (19M) [application/octet-stream]
Сохранение в: «Python-3.12.6.tar.xz»

Python-3.12.6.tar.x 100%[===================>]  19,49M  8,39MB/s    за 2,3s

2024-09-24 16:17:38 (8,39 MB/s) - «Python-3.12.6.tar.xz» сохранён [20434028/2043
4028]
```

```
File Edit > tions Buffers Tools Python Help
     Undo                    C-x u
LOG  Cut                     C-w
     Copy                    M-w
     Paste                   C-y
clas Paste from Kill Menu >
     Clear
     Select All              C-x h
     --                                                    | None = None, status_code: int | None = No\
     Search >                                                                                          \
ne): Replace >
     Go To >                 ESSAGE
     Bookmarks >
     --
     Fill                    ode or self.STATUS_CODE
     Text Properties >
    def details(self) -> dict[str, typing.Any]:
        return {
            'msg': self.message
        }


-UU-:%%--F1  core.py          1% L16     (Python) ----------------------------
Undo last operation
```

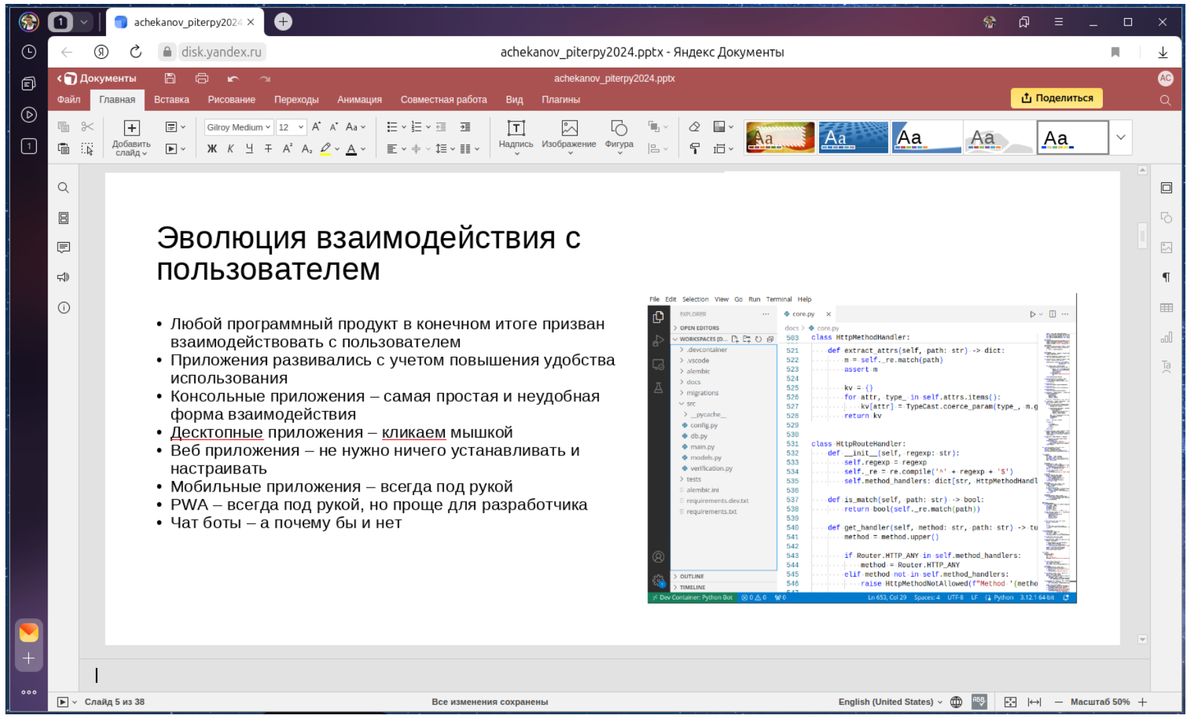
Эволюция взаимодействия с пользователем

- Любой программный продукт в конечном итоге призван взаимодействовать с пользователем
- Приложения развивались с учетом повышения удобства использования
- Консольные приложения – самая простая и неудобная форма взаимодействия
- Десктопные приложения – кликаем мышкой
- Веб приложения – не нужно ничего устанавливать и настраивать
- Мобильные приложения – всегда под рукой
- PWA – всегда под рукой, но проще для разработчика
- Чат-боты – а почему бы и нет



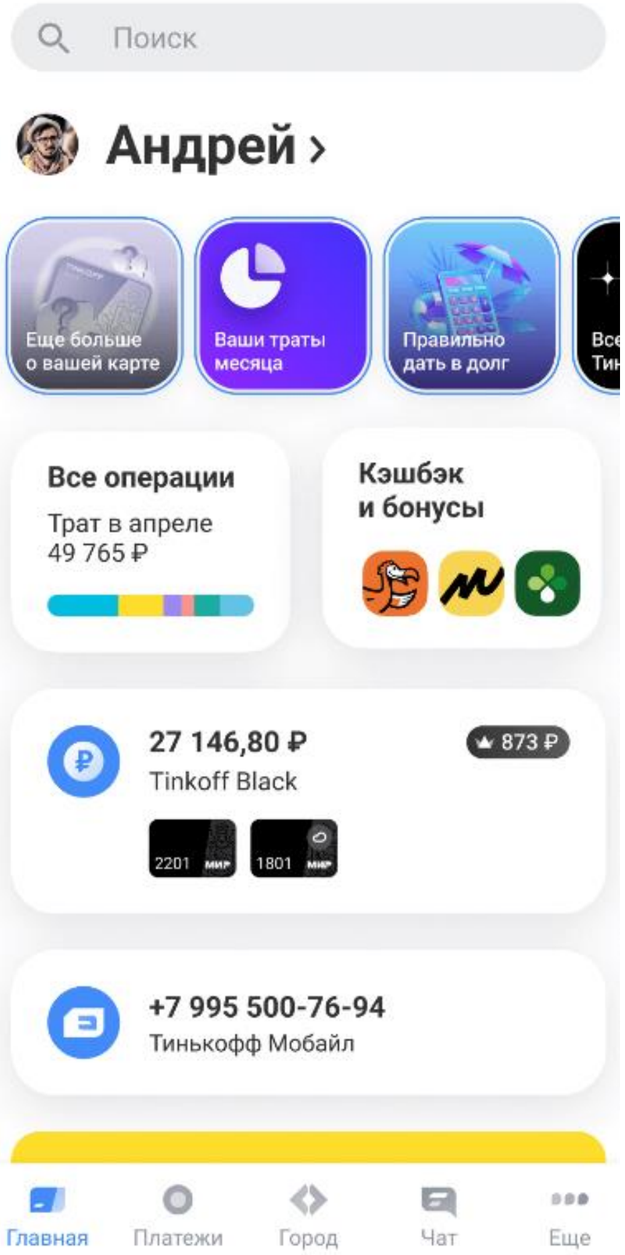
Эволюция взаимодействия с пользователем

- Любой программный продукт в конечном итоге призван взаимодействовать с пользователем
- Приложения развивались с учетом повышения удобства использования
- Консольные приложения – самая простая и неудобная форма взаимодействия
- Десктопные приложения – кликаем мышкой
- Веб приложения – не нужно ничего устанавливать и настраивать
- Мобильные приложения – всегда под рукой
- PWA – всегда под рукой, но проще для разработчика
- Чат-боты – а почему бы и нет



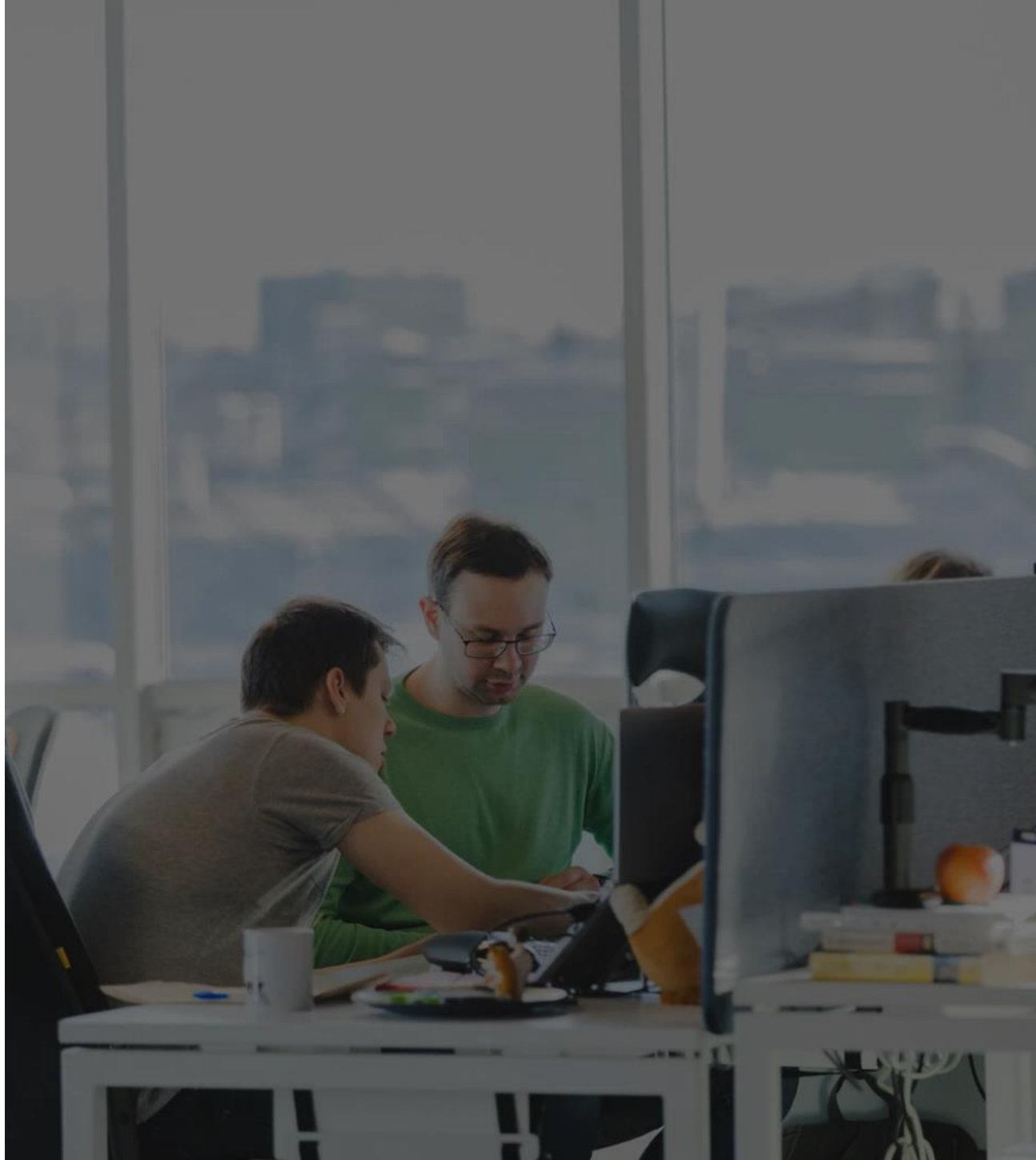
Эволюция взаимодействия с пользователем

- Любой программный продукт в конечном итоге призван взаимодействовать с пользователем
- Приложения развивались с учетом повышения удобства использования
- Консольные приложения – самая простая и неудобная форма взаимодействия
- Десктопные приложения – кликаем мышкой
- Веб приложения – не нужно ничего устанавливать и настраивать
- Мобильные приложения – всегда под рукой
- PWA – всегда под рукой, но проще для разработчика
- Чат-боты – а почему бы и нет



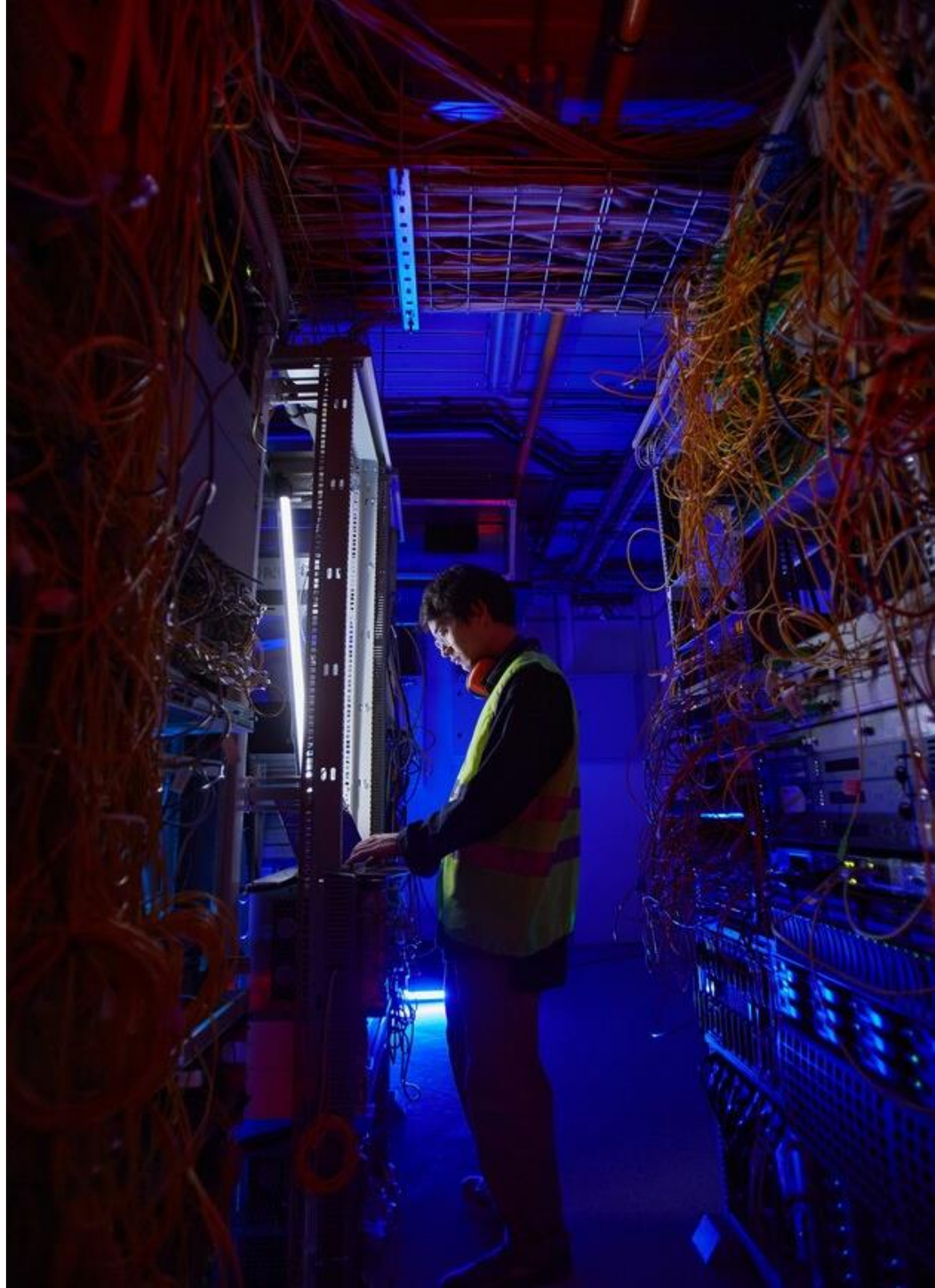
02

История одного продукта: как мы пришли к идее чат-бота



Чат-бот как полноценное приложение

- Телеграм вроде у всех есть
- К интерфейсу все привыкли
- Меньше затрат на UI
- Десктоп, мобилки
- Много возможностей для кастомизации
  - Сообщения
  - Инлайн кнопки
  - Маркап в сообщениях
  - Webapps
- ...
- А почему бы и нет?



История одного продукта



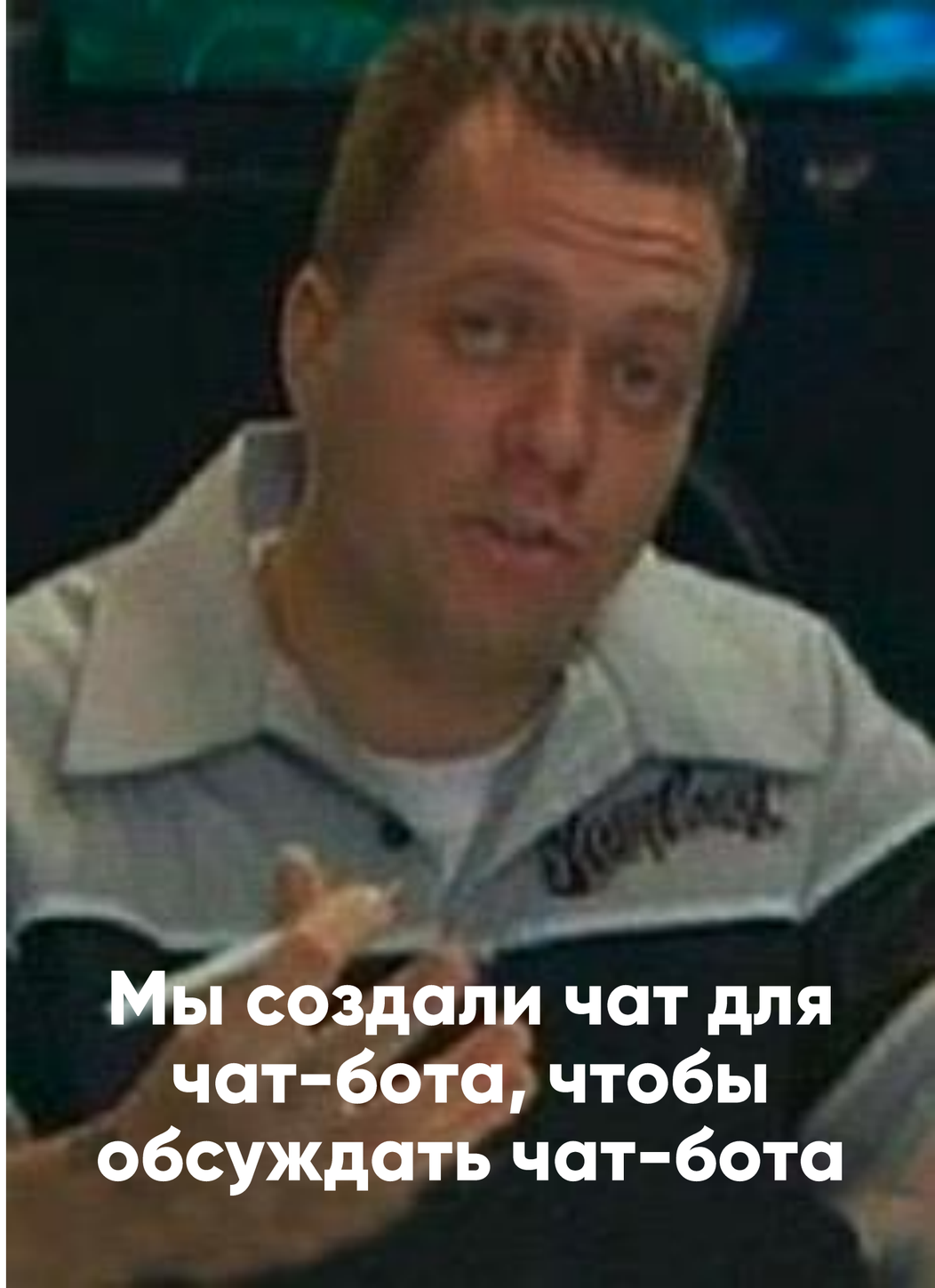
Объединить ряд внутренних сервисов в едином интерфейсе

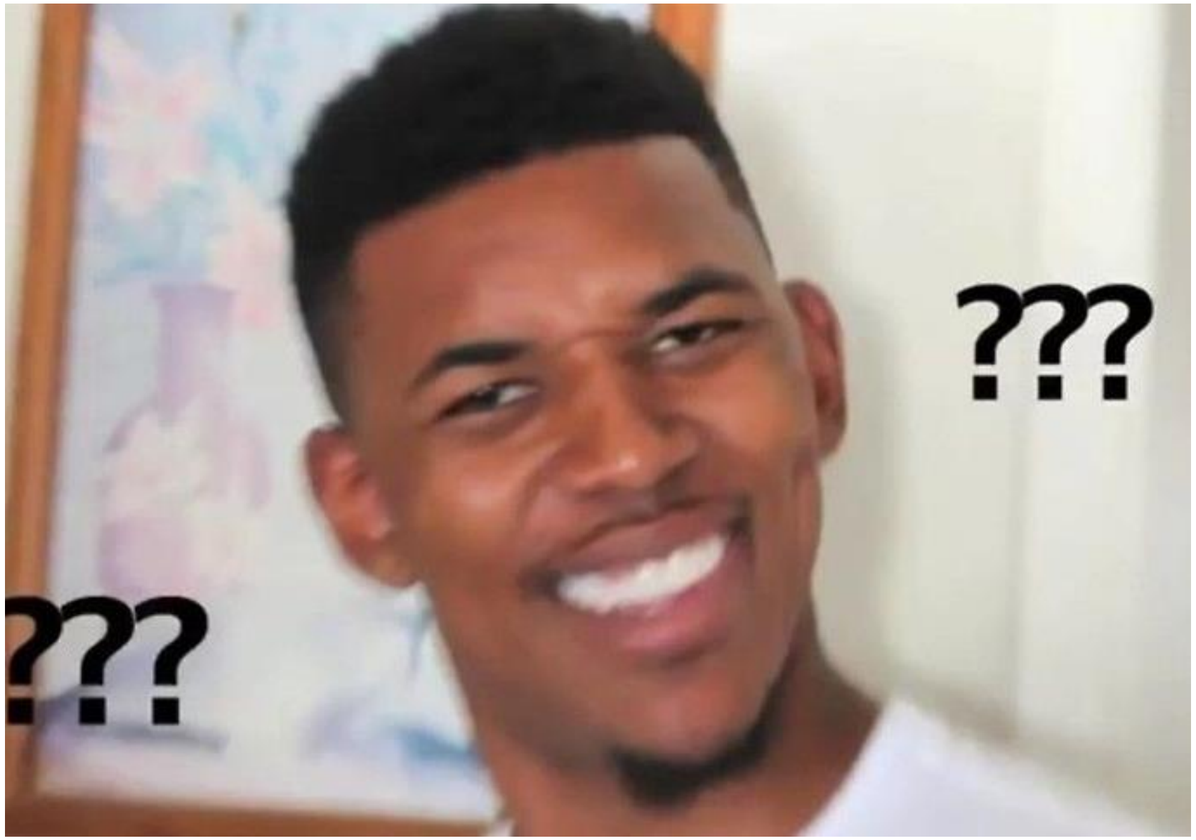
Переработать и, возможно, слить воедино уже существующие боты

Сделать так, чтобы работало шустро и надежно

Проблемы

- В сети полно информации о том как сделать чат-бот
  - Находим BotFather
  - Отвечаем на пару вопросов
  - Берем ключи
  - Стучимся локально в getUpdates
  - Обрабатываем логику
  - Стучимся в апи sendMessage
  - Читаем рекламную интеграцию в статье где все это можно быстро и дешево захостить
- Очень мало информации про конкретные задачи и их решения
- Множество различных фреймворков (мы написали еще один)
- И вообще не очень понятно как сделать энтерпрайз бота





Энтерпрайз бот?

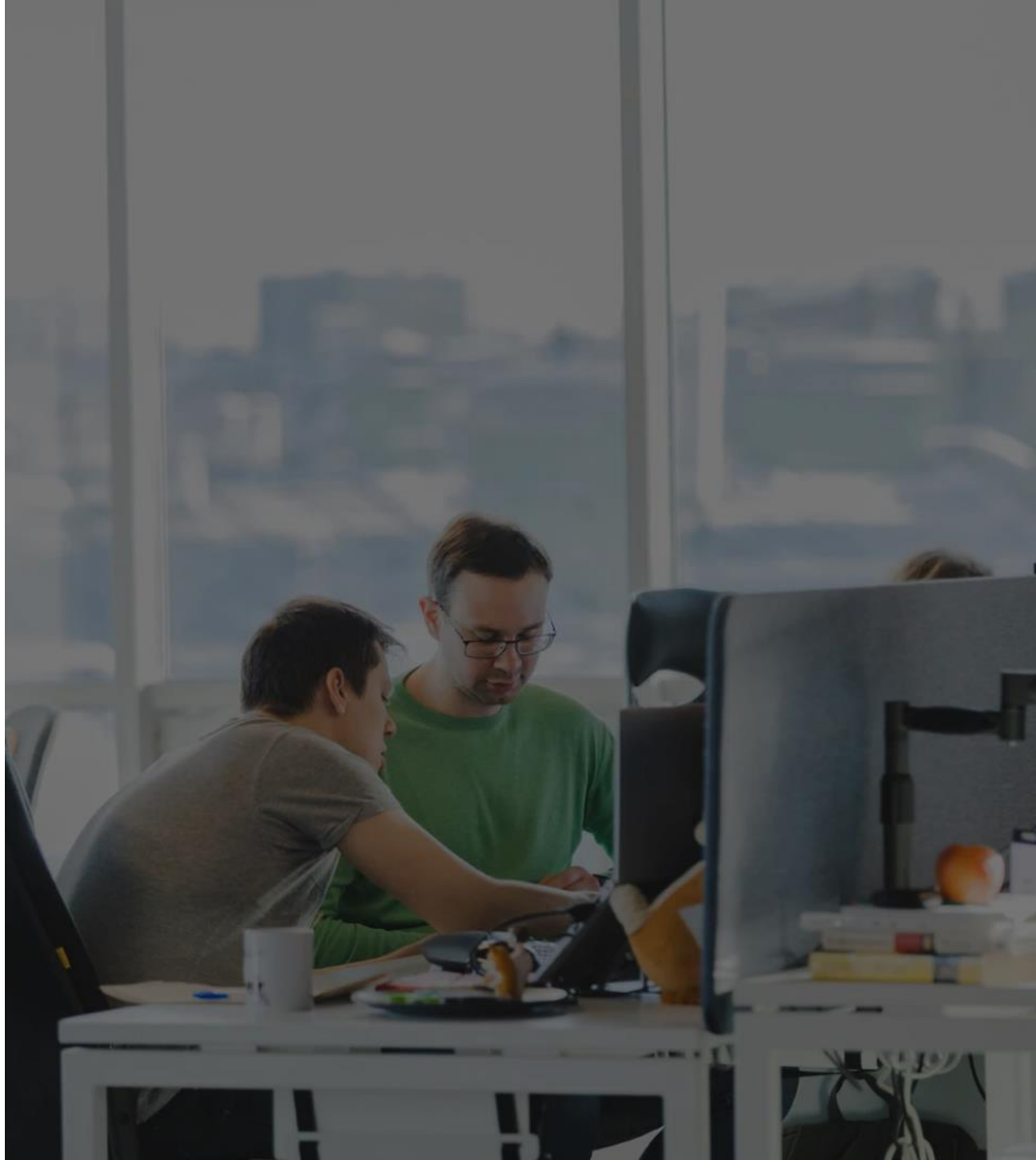
Энтерпрайз бот!

- Мы как компания работаем в секторе B2B
- Заказчик крупная компания
- Над ботом работает команда, а не один человек
- Нужны различные енвайры
- Нужны какие-то хорошие практики и примеры процессов разработки
- Паттерны

Однако этой информации маловато...

03

Как мы внедряли чат-бота в инфрастуктуру компании

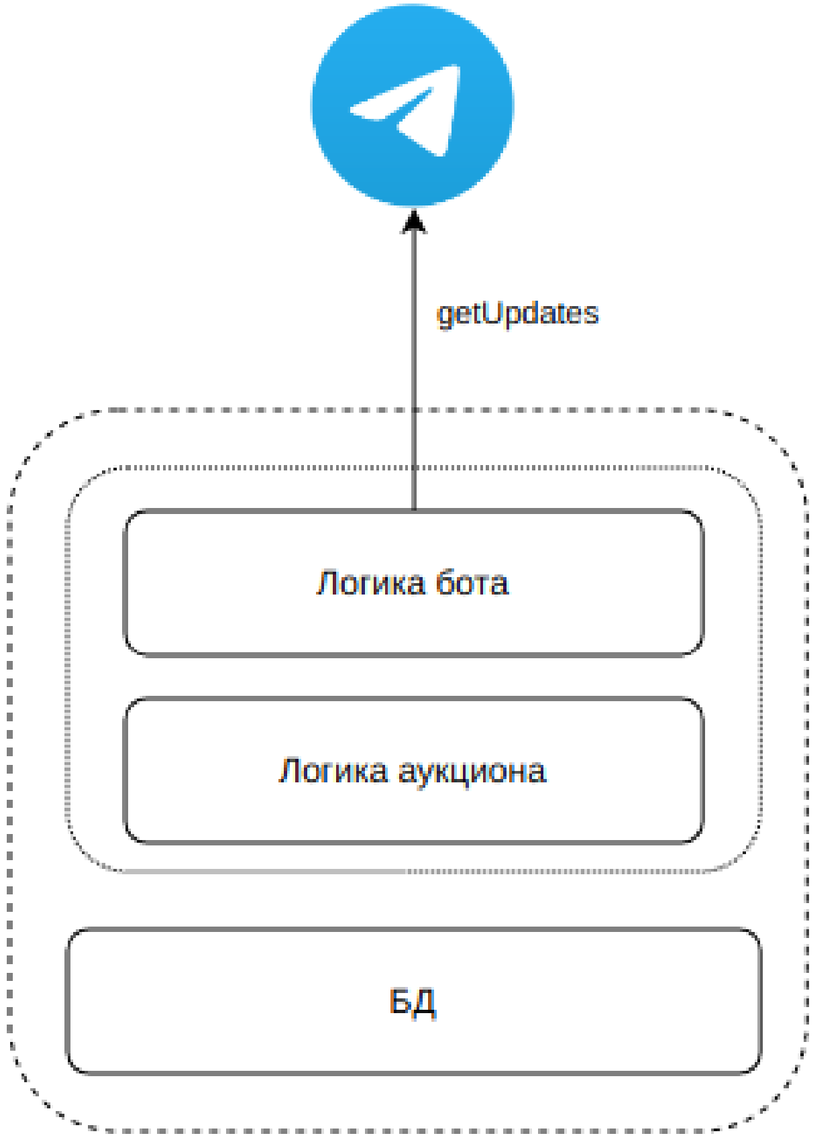


Аудит текущего бота

- Бот - внутренний аукцион
  - Парковочных мест
  - Старых комплектующих
  - Пользовательские
- Аукцион простой, открытый, на повышение ставки
- На горячих лотах тормозит
- Иногда отваливается (Не страшно, но неприятно)
- Было бы здорово иметь админку не в боте а на сайте

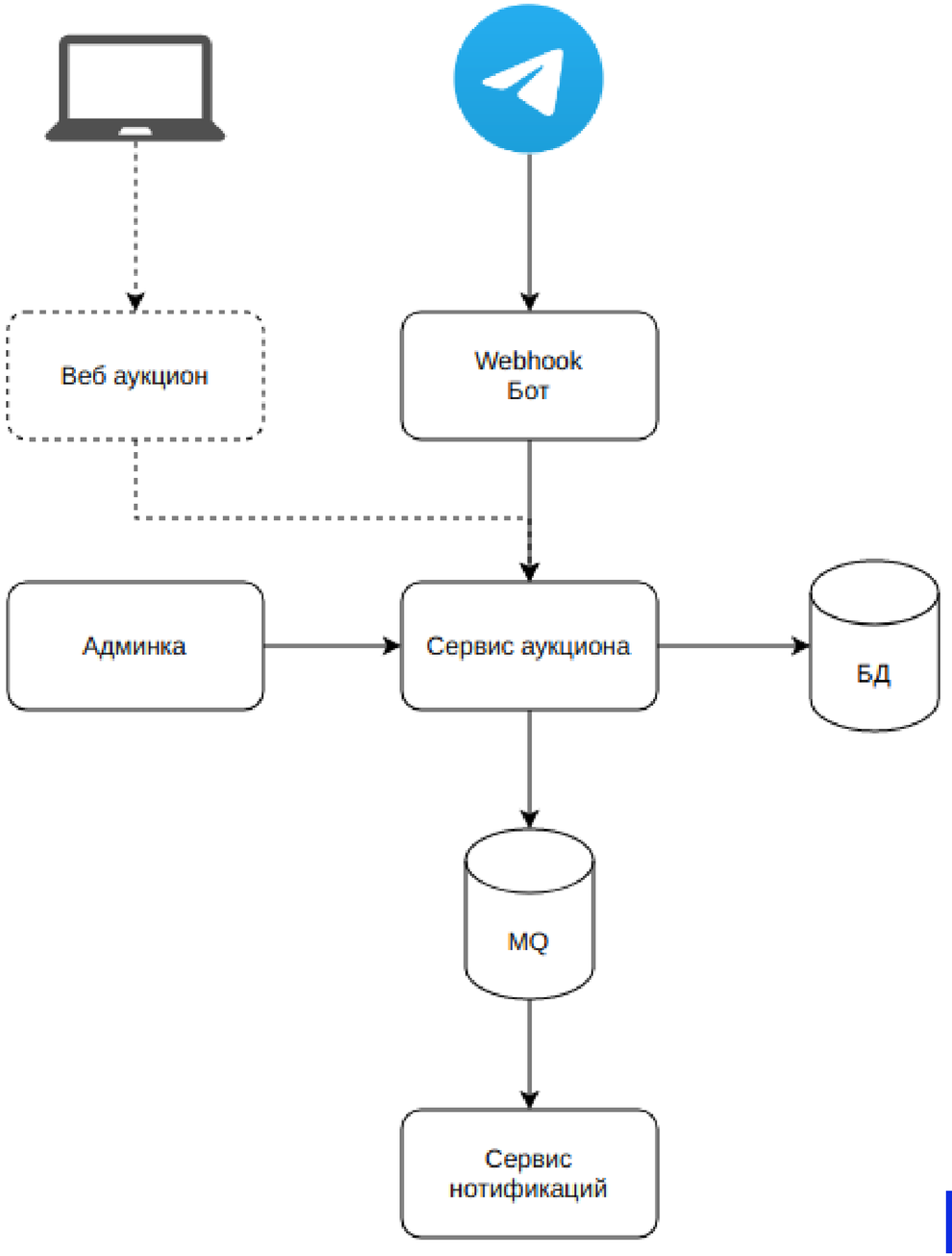
Итоги аудита

- Бот работает в пул режиме
- Бот работает на одном инстансе
- Вся необходимая информация находится на этой же виртуалке
- В текущем виде как-то масштабировать его невозможно



Шаги по изменению

- Логика не должна быть в коде самого бота
- Бот – всего лишь presentation layer
- Бот должен использовать механизм webhooks
- Операции над лотом должны быть атомарными



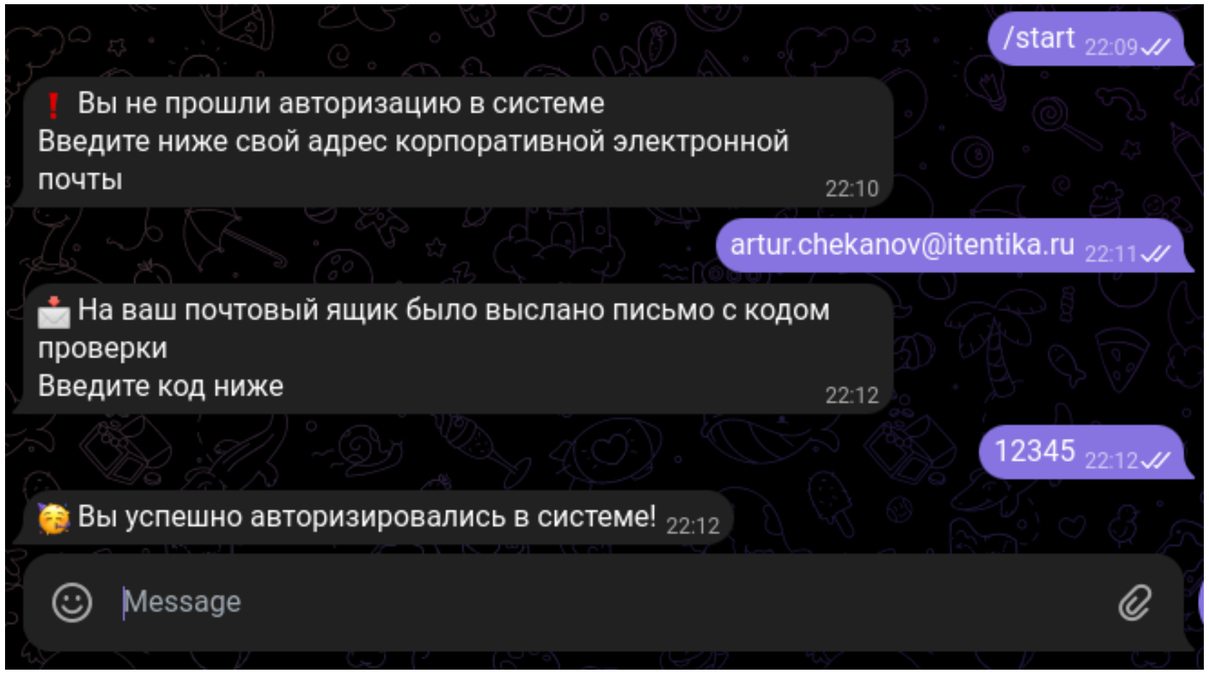
Авторизация бота

- Бот предназначен для внутреннего пользования
- Когда учетная запись заводится для сотрудника он может добавить свой телеграм ник
- Пользователи с другим ником игнорируются ботом
- Есть некоторые проблемы
  - В нике можно ошибиться
  - Ник можно поменять
  - Можно сделать логин прямо в боте

```
sender = msg['message']['from']
stmt = select(User).where(User.tg_username == sender['username'])
result = session.execute(stmt)

user = result.scalars().one_or_none()
if user is None:
    tg_request('sendMessage', chat_id=chat_id, text=UNKNOWN_USER_MESSAGE, parse_mode="MarkdownV2")
```

Авторизация бота

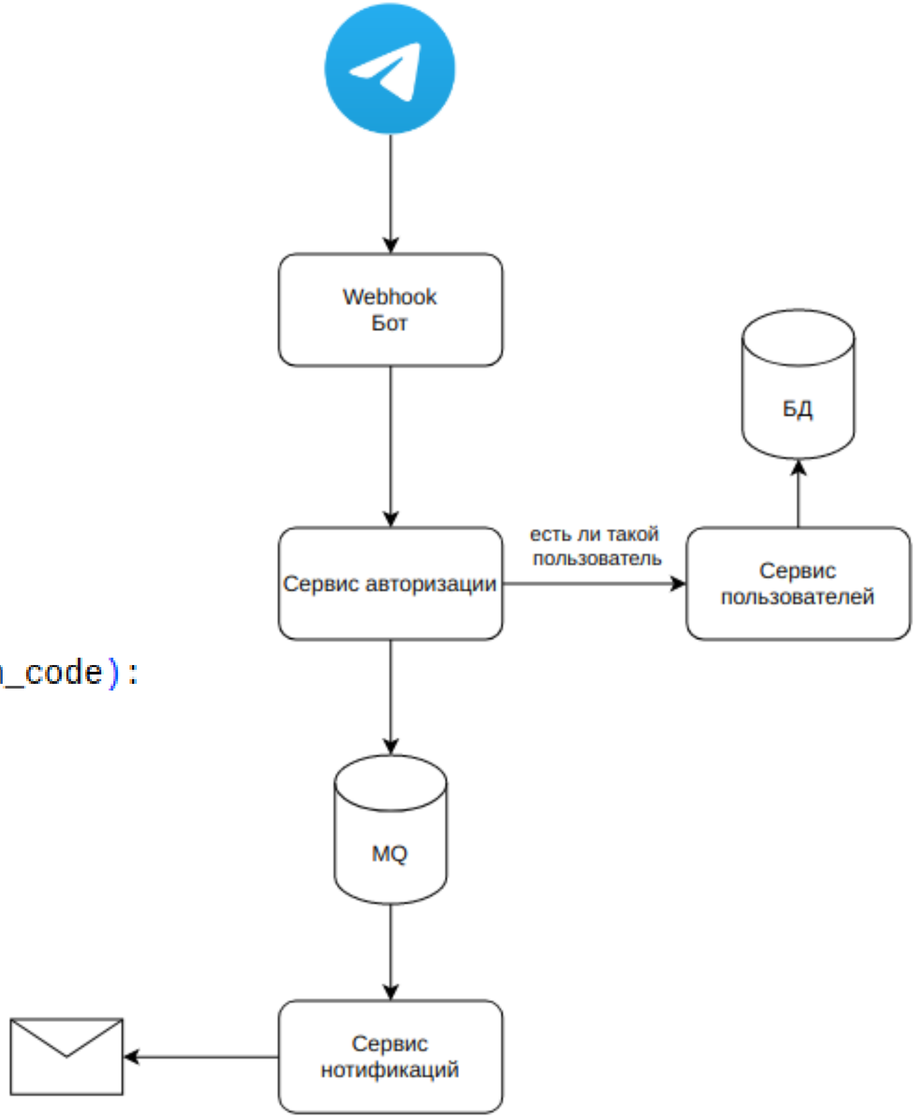
- Можно использовать подтверждение через электронную почту
- Почта должна быть корпоративая
- Плюсом идет автоматическая привязка тг к аккаунту
- Плюсом идет чистка рабочих и не очень чатов при увольнении сотрудника



Авторизация бота

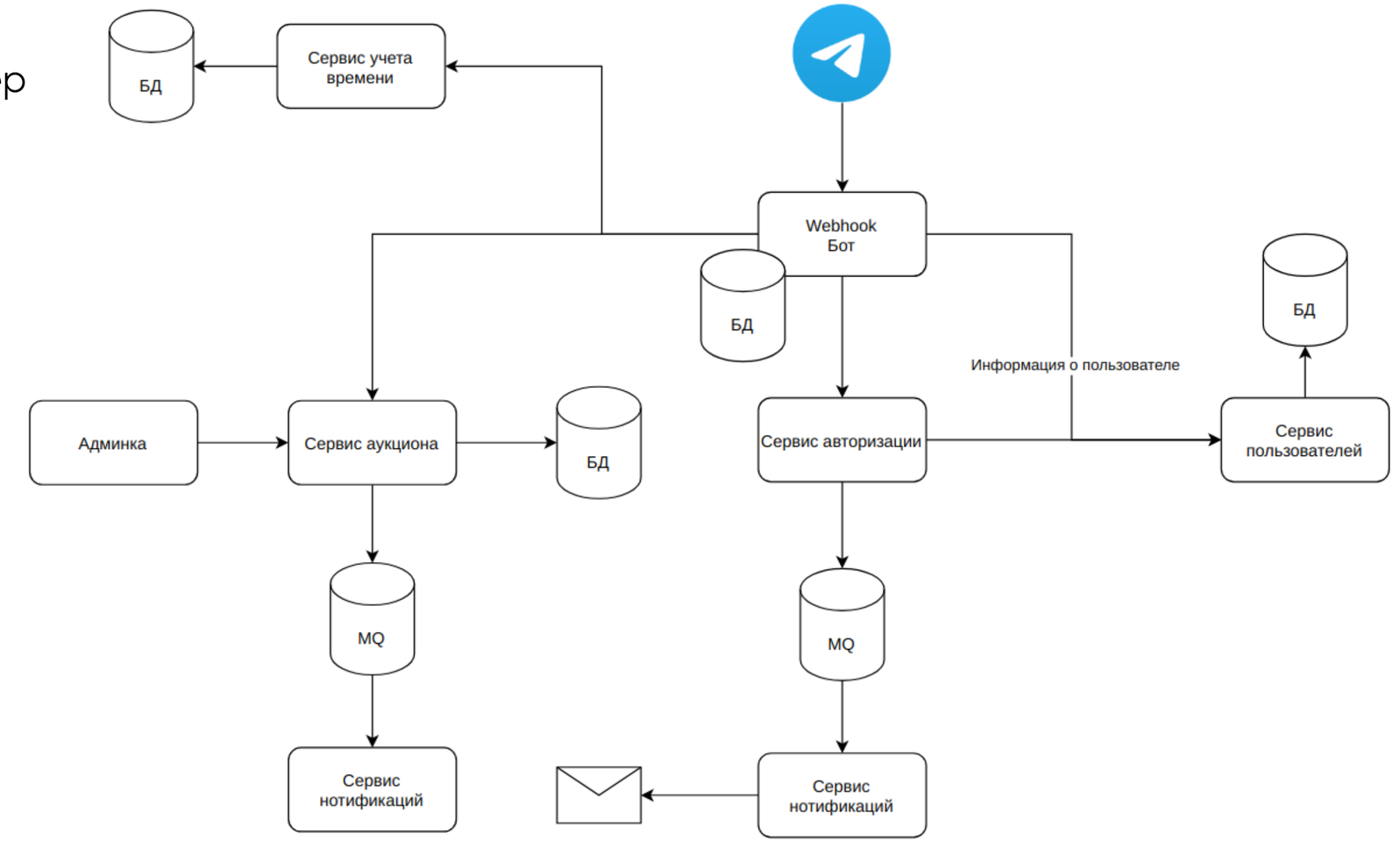
```python
def _generate_verification_code():
    secret_code = []
    for _ in range(settings.verification_code_length):
        secret_code.append(str(secrets.randbelow(10)))
    return ''.join(secret_code)
```

```python
def _verify_code(verification_code: str, verification_id: str):
    verification_attempt_increase(verification_id)
    code = _get_code(verification_id)
    if not code or not secrets.compare_digest(code.code, verification_code):
        raise APIError(APIError.Error('Invalid code',
                                      location='body', parameter='verification'))
```



Развитие. Добавление новых фич

- Информация о сотруднике
  - Имя, почта, линейный менеджер
- Логирование времени
- Общая информация о компании
  - Полезные ссылки
  - График выходных



Бронирование митинг румов

- Митинг румы никак не бронировались
- Если нужно было забронировать митинг рум на нем приклеивался стикер со временем бронирования
- Но обычно все просто искали свободный
- Было решено поставить неинтерактивные дисплеи с расписанием и текущим статусом
- В качестве начинки был выбран Arduino и трехцветные e-ink дисплеи

Добавление новых фич

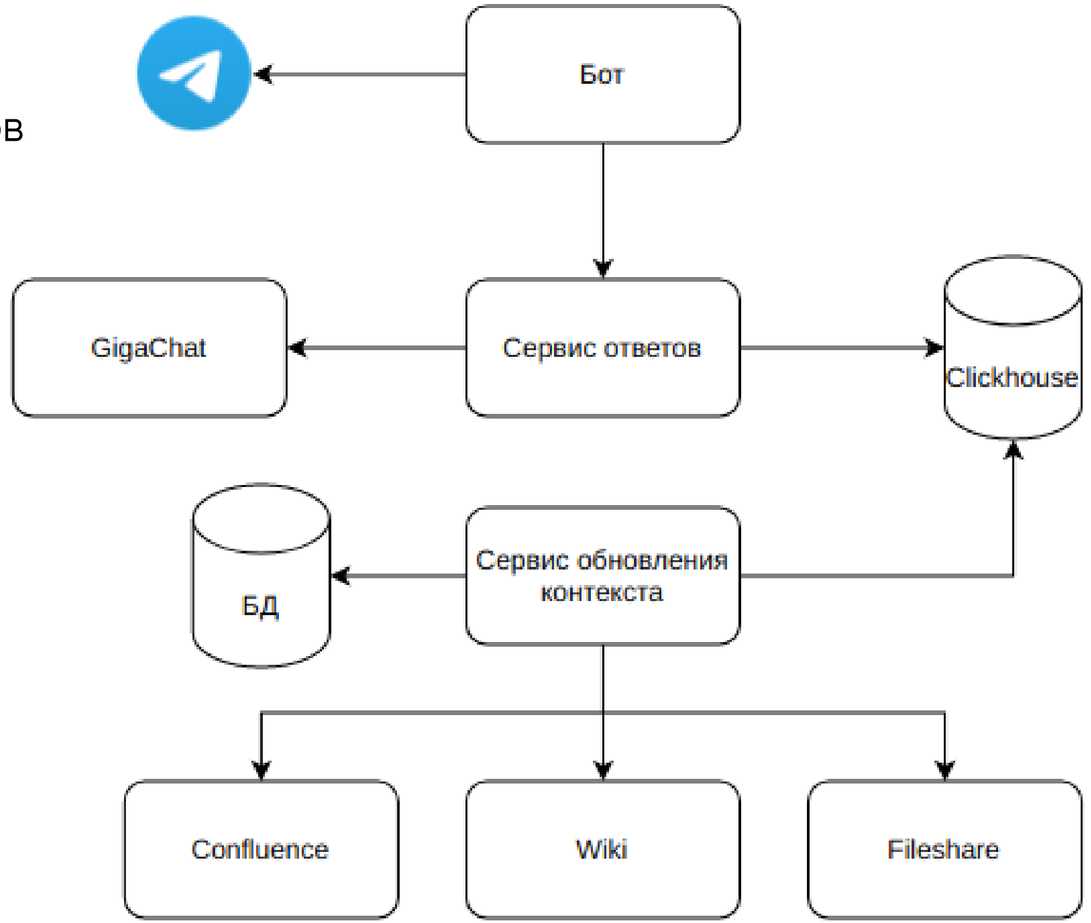
- Базовый аукцион практически не имел стейта
- С появлением новых фич, нужно было как-то отслеживать стейт
- Обчно бекенд stateless
- Стейт обычно на фронте
- Но не в боте

```python
@on_state(BookingOfficeSelectState)
def process_booking(user_state: UserState,
                    message_state: MessageState,
                    message: TgChatMessage):
    ...
    command = state.llm.get_command(message)
    if command is None:
        ...
        message.answer("Нам неизвестен такой офис, попробуйте еще раз")
        state = state.next(state.prev_state)
        ...

    match command:
        ...
        case 'spb_office':
            ...
            state = state.next(BookingRoomSelectState(office=command))
```

```python
def dispatch(message: TgMessage):
    ...
    user_state: UserState = get_current_state(message)

    match message:
        case TgChatMessage():
            ...
            user_state.process(message)
            ...
            dispatch_user_state(user_state, message)
        case TgCallbackQuery():
            ...
            message_state: MessageState = get_message_state(message)
            ...
            dispatch_message_state(message_state, message)
            ...
```

Добавление боту интелекта. Исскуственного

- Куда же без него
- В компании существует большое количество документов во внутреннем Confluence, wiki, файлшаре и т.д.
- Бот должен уметь отвечать на вопросы с учетом внутренней информации
- Использовали RAG
  - Разбивали статьи на embeddings
  - Маркировали, где-то автоматически, где-то руками
  - Доступ по ABAC



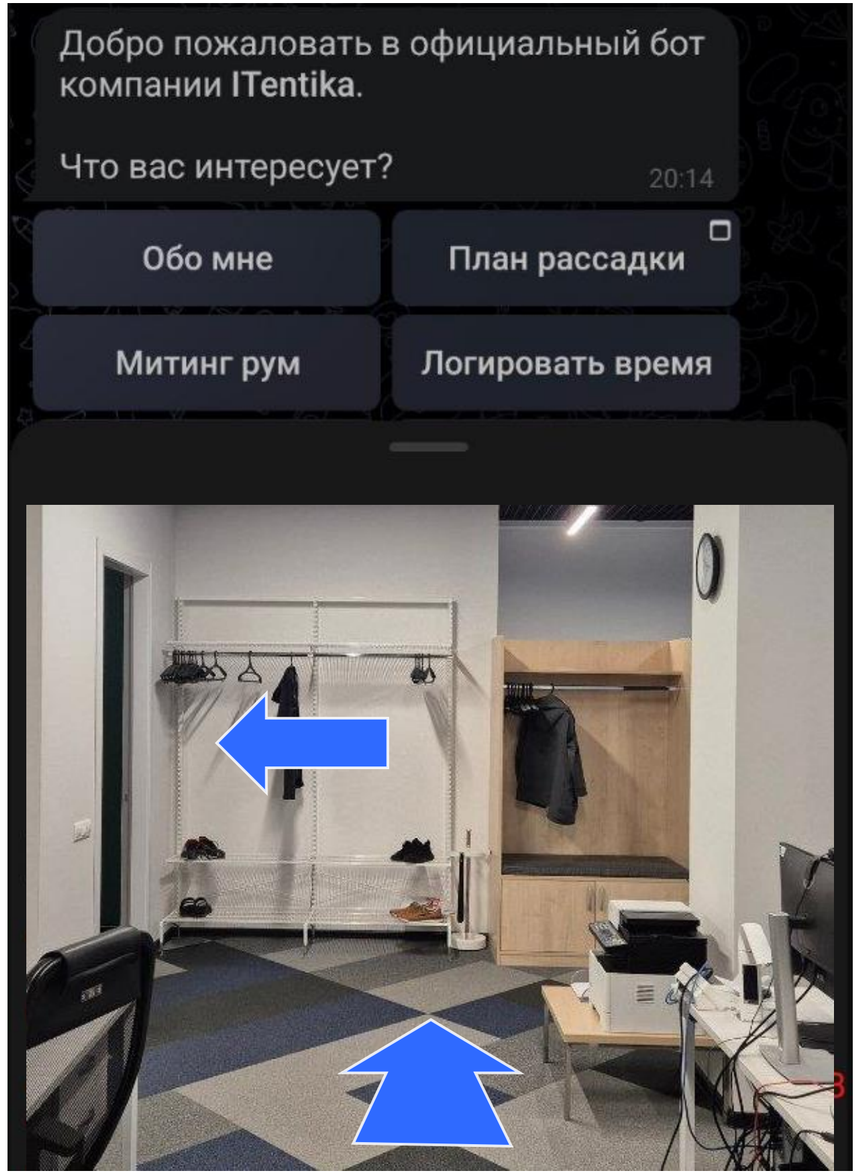
RAG

1. Проверяем источники данных на обновления
2. Новые статьи разбивает на параграфы и считаем по ним емебединги
3. Запрос также получает эмбединг, сначала поиск по кешу
4. Поиск по соответсвиям
5. Фильтрация по атрибутам ABAC
6. Наполнение запросы LLM нужным контекстом

```python
def rag_answer(question):
    embedding = get_embedding(question)
    results = clickhouse.query(
        """
        select
            content, access_attributes, L2Distance(%(embedding)s, embedding) as score
        from kb
        order by score ASC
        limit %(limit)s
        """, parameters={'embedding': embedding, 'limit': limit})

    rag = _filter_abac(user_attributes,
        (content, attrs for content, attrs, _ in results.result_rows))

    rag = [content for content, _ in results.result_rows]
    return get_answer(question, rag)
```

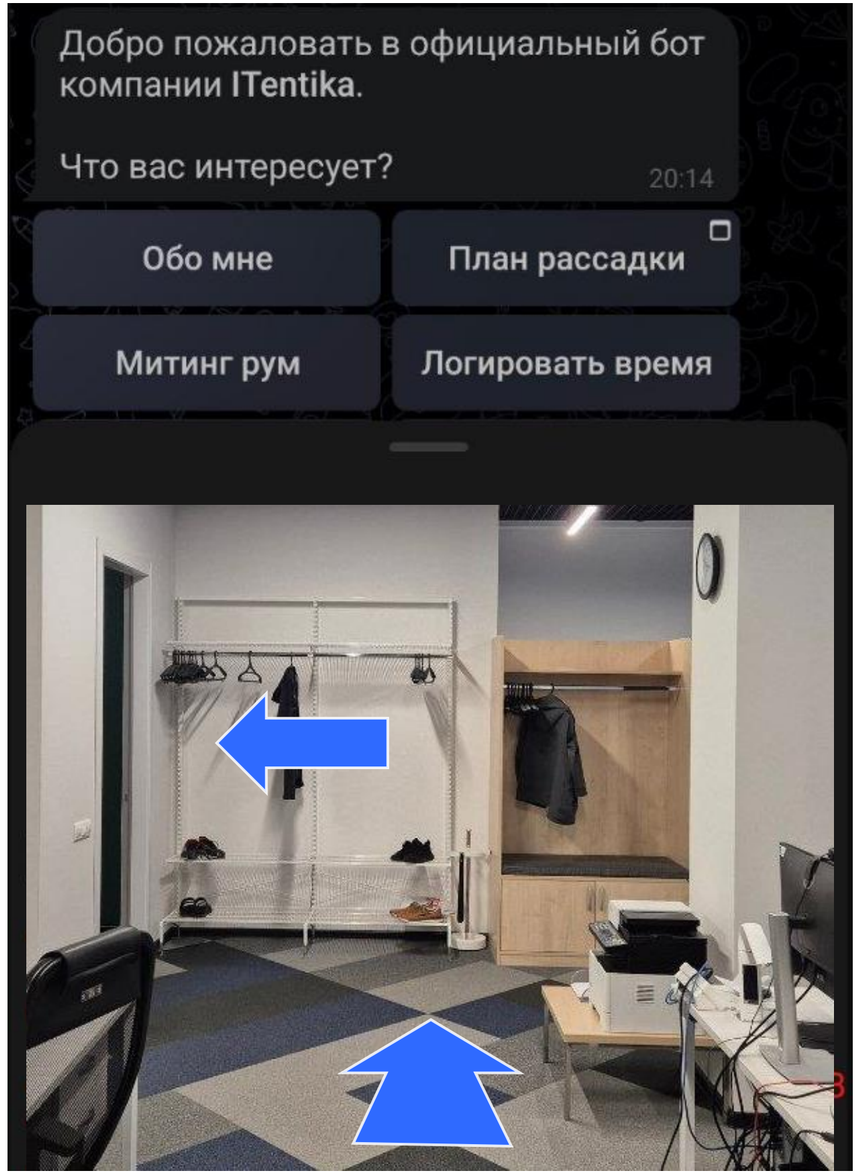
Веб аппы. План рассадки

- Самая амбициозная часть проекта
- Все еще в разработке
- Идея – а что если добавить AR позволяющий ориентироваться по офису?
  - Реализации подобной технологии уже начали появлятся на рынке
  - Ориентация в магазине
  - Ориентация по складским помещениям
- GPS не пойдет
- Без блютус маячков и прочих хардварных решений
- Максимум с визуальными маркерами
- Желательно путем чистого CV (пока не совсем получается)
- Но как это сделать через бота?



Веб аппы. Что это??

- Телеграм имеет встроенный браузер
- Бот может показывать веб страницу по
  - Нажатию на inline кнопку
  - По нажатию на ссылку
  - В меню
  - ...
- Странице передается нужный контекст
- Возможности ограничены только возможностями браузера



WebApps

- Легко встроить

```
request('sendMessage', chat_id=msg['message']['chat']['id'],
        text=START_MESSAGE,
        parse_mode="MarkdownV2",
        reply_markup=json.dumps({
    'inline_keyboard': [
        [
        ...
        {
            "text": "План рассадки",
            "web_app": {"url": settings.seating_plan_webapp_url},

        }
        ...
        ],
        ...
    ]
}))
```

WebApps

- Нужный контекст

```
<script src="https://telegram.org/js/telegram-web-app.js"></script>

data = {
    'init': window.Telegram.WebApp.initData
}
const response = await fetch('/auth', {
    method: 'POST',
    body: JSON.stringify(data)
});
```

```
secret_key = hmac.new("WebAppData".encode(),
                      settings.bot_token.encode(),
                      hashlib.sha256).digest()
data_check = hmac.new(secret_key,
                      init_data.encode(),
                      hashlib.sha256).digest()
if hmac.compare_digest(data_check, data_hash):
```

WebApps

- Возможности ограничены возможностями браузера

```
navigator.mediaDevices
    .getUserMedia({'video': { facingMode: "environment" }, 'audio': false})
    .then((stream) => {
        video.srcObject = stream;
        video.load();
        video.play();
    })
    .catch((err) => {log(err)});
```

```
const sensor = new AbsoluteOrientationSensor();
sensor.addEventListener("reading", (e) => {
    // ...
});
sensor.addEventListener("error", (event) => {
    // ...
});
sensor.start();
```

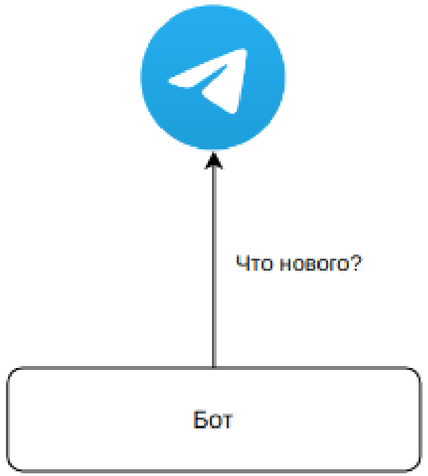
WebApps

- Есть нюансы

`acceleration`	✓ 31	✓ 12	✓ 6	✓ 18	✓ 17	✓ 31	✓ 6	✓ 18	✓ 4.2	✓ 2.0	✓ 4.4.3
`accelerationIncludingGravity`	✓ 31	✓ 12	✓ 6	✓ 18	✓ 17	✓ 31	✓ 6	✓ 18	✓ 4.2	✓ 2.0	✓ 4.4.3
`interval`	✓ 31	✓ 12	✓ 6	✓ 18	✓ 17	✓ 31	✓ 6	✓ 18	✓ 4.2	✓ 2.0	✓ 4.4.3
`requestPermission()` static method	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✓ 14.5	✗ No	✗ No

Как разрабатывать локально?

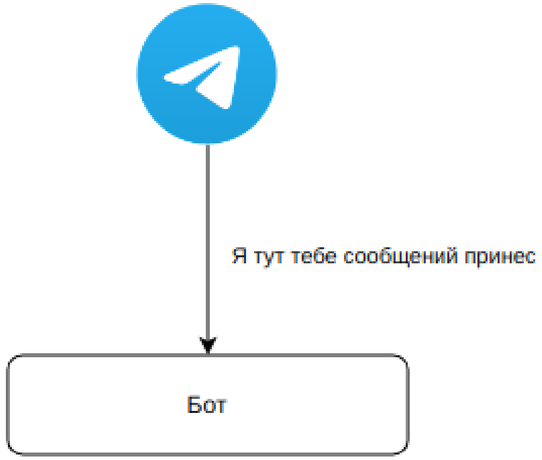
Пулл метод



+ Быстро и просто
+ Можно локально
+ Подходит для тестирования

– Трудно масштабировать
– Уже прочитанные сообщения нужно обрабатывать самому

Пуш метод



+ Хорошо масштабируется
+ Чуть экономим CPU циклы и трафик

– Труднее тестировать локально
– Нужен сервер
– Нужно доменное имя
– Нужен TLS

Как разрабатывать локально?

- Поддерживаем оба режима
- По умолчанию в __name__ == '__main__' делаем get Updates
- В енвах запускаем uvicorn main:app
- У каждого разработчика свой тестовый бот
- Различные фича свитчи под пользователя/версию приложения
    - по принципам постоянно интеграции

```
if settings.tg_update_mode == MODE_PULL:
    ...
    if tg_is_webhook():
        tg_remove_webhook()
    tg_poll_start(update_callback)
else:
    ...
    if not tg_is_webhook():
        tg_set_webhook()
    tg_app_start(update_callback)
```


Как разрабатывать локально? WebApps 1/2

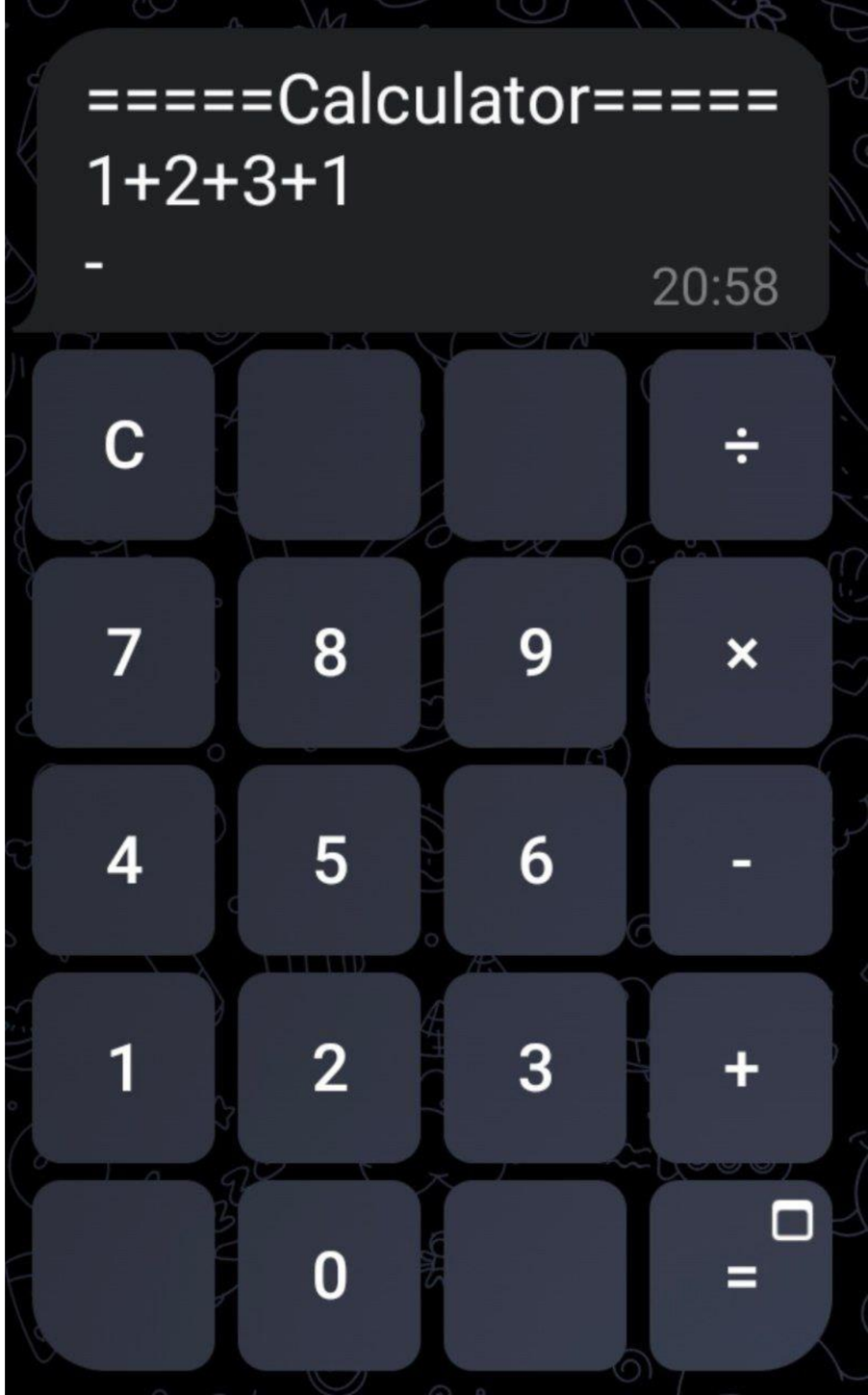
- WebApp хочет какой-то бекенд иметь
- Основная проблема поднять локальный сервер чтобы он был доступен
- Можно работать из веб телеграма и отправлять все запросы на localhost
- Но что делать с камерой веб аппа
    - Создавать виртуальную камеру на PC

Как разрабатывать локально? WebApps 2/2

- Делать локально в веб браузере
- Постоянно деплоить
- Править руками на сервере
- Делать remote по ssh
- Ngrok но это серый метод

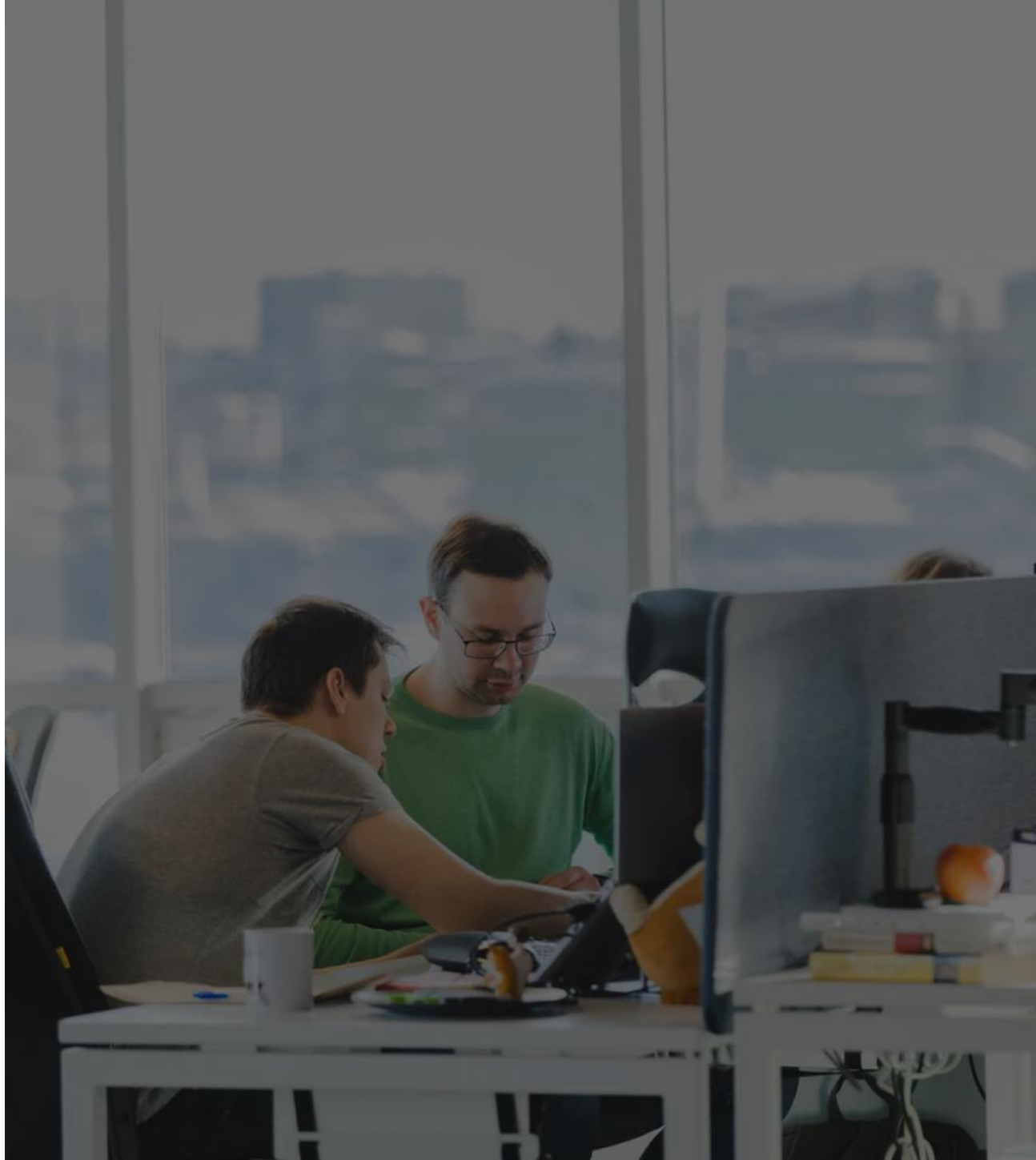
Демо-приложение. Калькулятор

- Калькулятор в тг боте
- С кнопками
- В режиме webhook
- Без приоритета операций
- Без скобочек



04

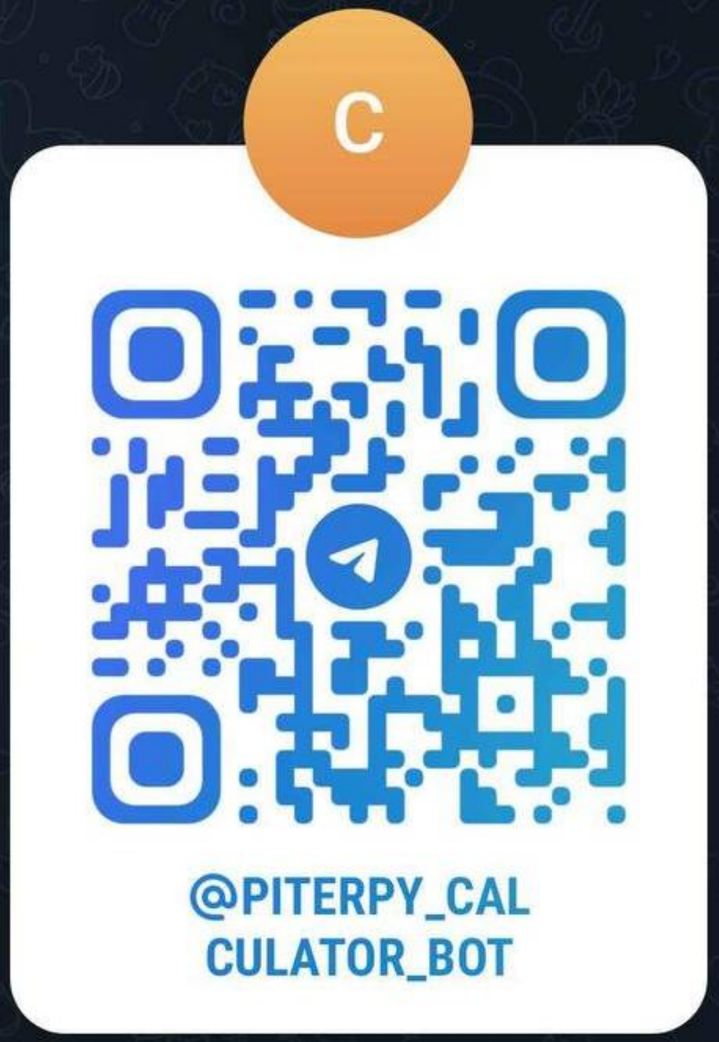
Выводы



Что мы получили

- Самый настоящий мегабот
- Проект объединил в себе самые разные задачи от IoT до AI
- Простой и удобный инструмент который всегда под рукой
- Доказательство того, что бот может во многом заменить мобильные и веб приложения

Это не призыв переносить все приложения на бота.Но посмотреть в их сторону точно стоит!



Контакты

Телефон

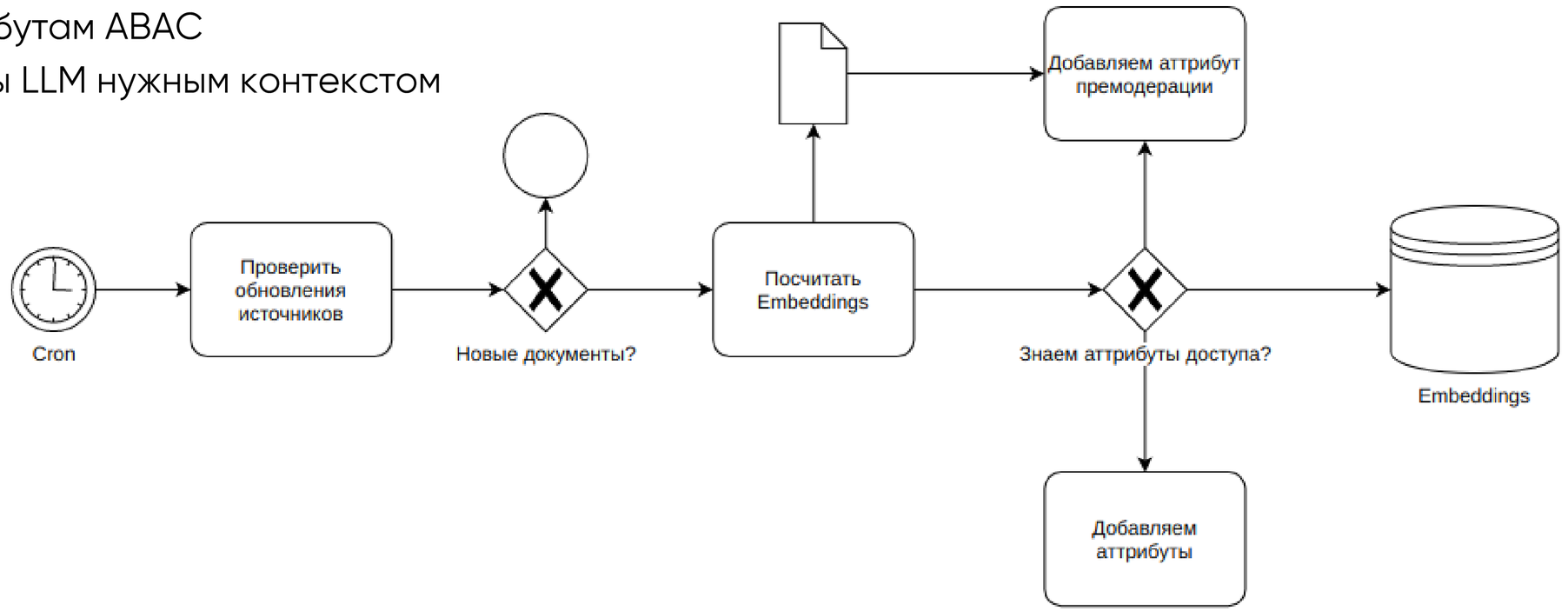
artur.chekanov@itentika.ru

Наш сайт

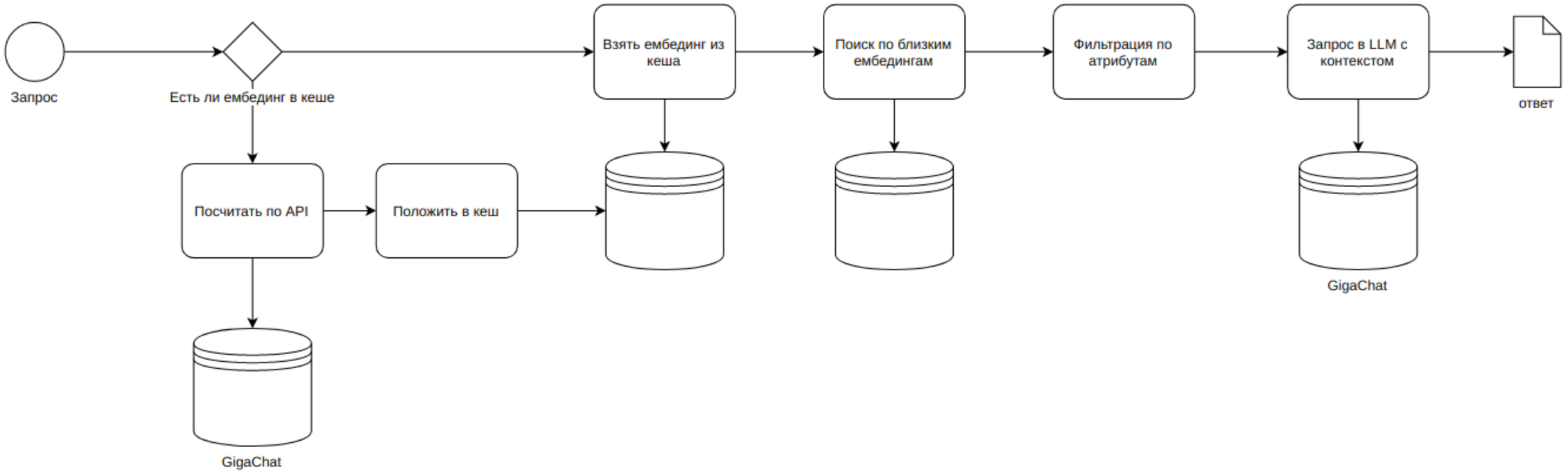
itentika.ru

RAG

- Проверяем источники данных на обновления
- Новые статьи разбивает на параграфы и считаем по ним ембединги
- Запрос также получает эмбединг, сначала поиск по кешу
- Поиск по соответсвиям
- Фильтрация по атрибутам ABAC
- Наполнение запросы LLM нужным контекстом



RAG


Запрос
Есть ли ембединг в кеше
Взять ембединг из кеша
Поиск по близким ембедингам
Фильтрация по атрибутам
Запрос в LLM с контекстом
ответ
Посчитать по API
Положить в кеш
GigaChat
GigaChat