FastAPI: создание сервиса для инференса ML моделей

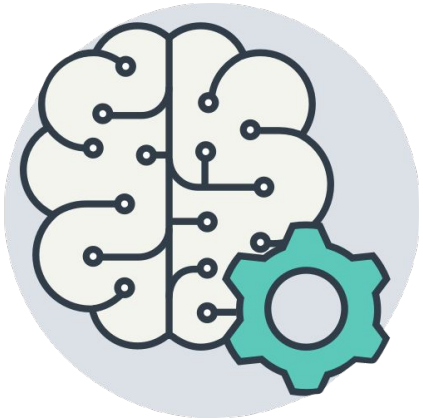
Практический курс по ML
для Райффайзен банка

Садртдинов Ильдус

25.05.2023

План

- Что такое API и зачем оно нужно для ML сервисов?
- Библиотека FastAPI, почему именно она?
- Практика: создаем свой сервис, оборачивая ML модели в FastAPI

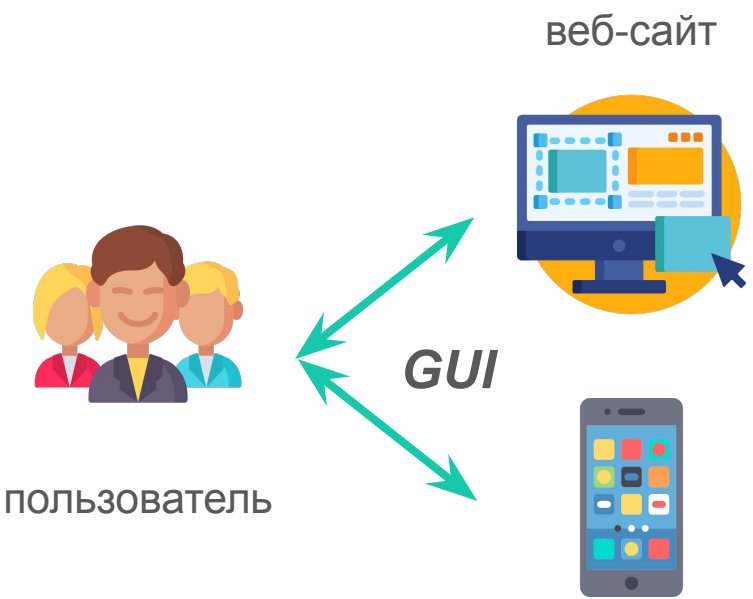


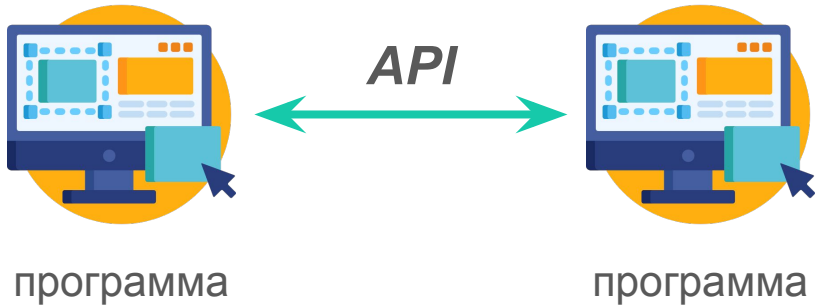
FastAPI

Что такое API?

- API – Application Programming Interface
- Некоторый свод правил, по которому мы можем общаться с приложением/программой

API vs GUI

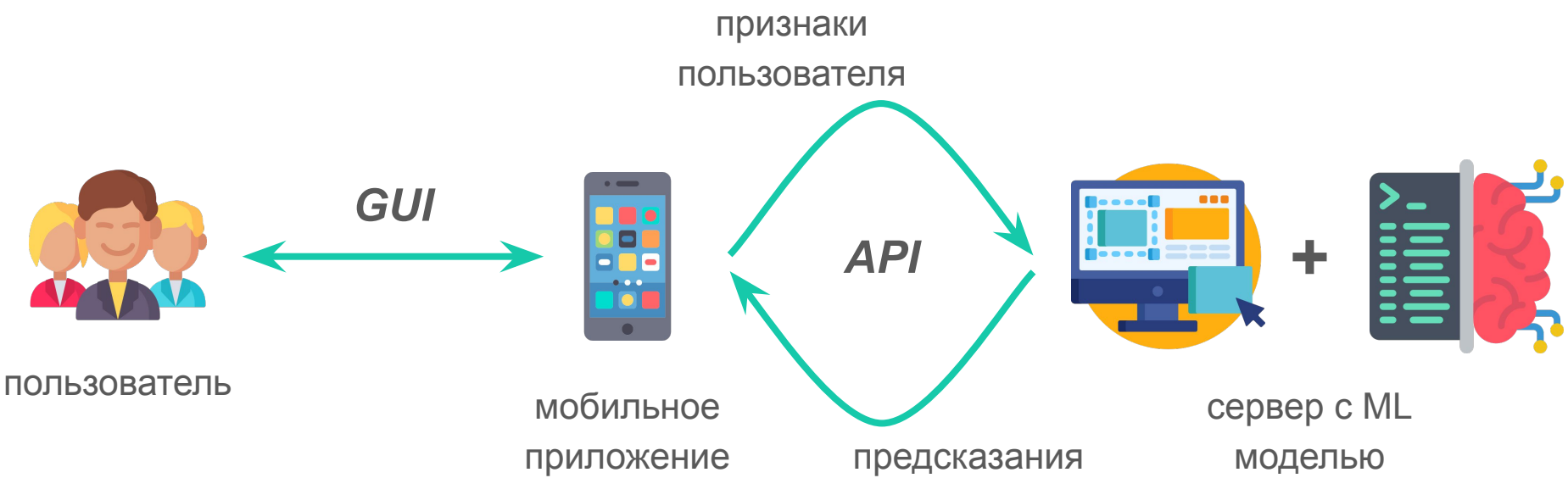




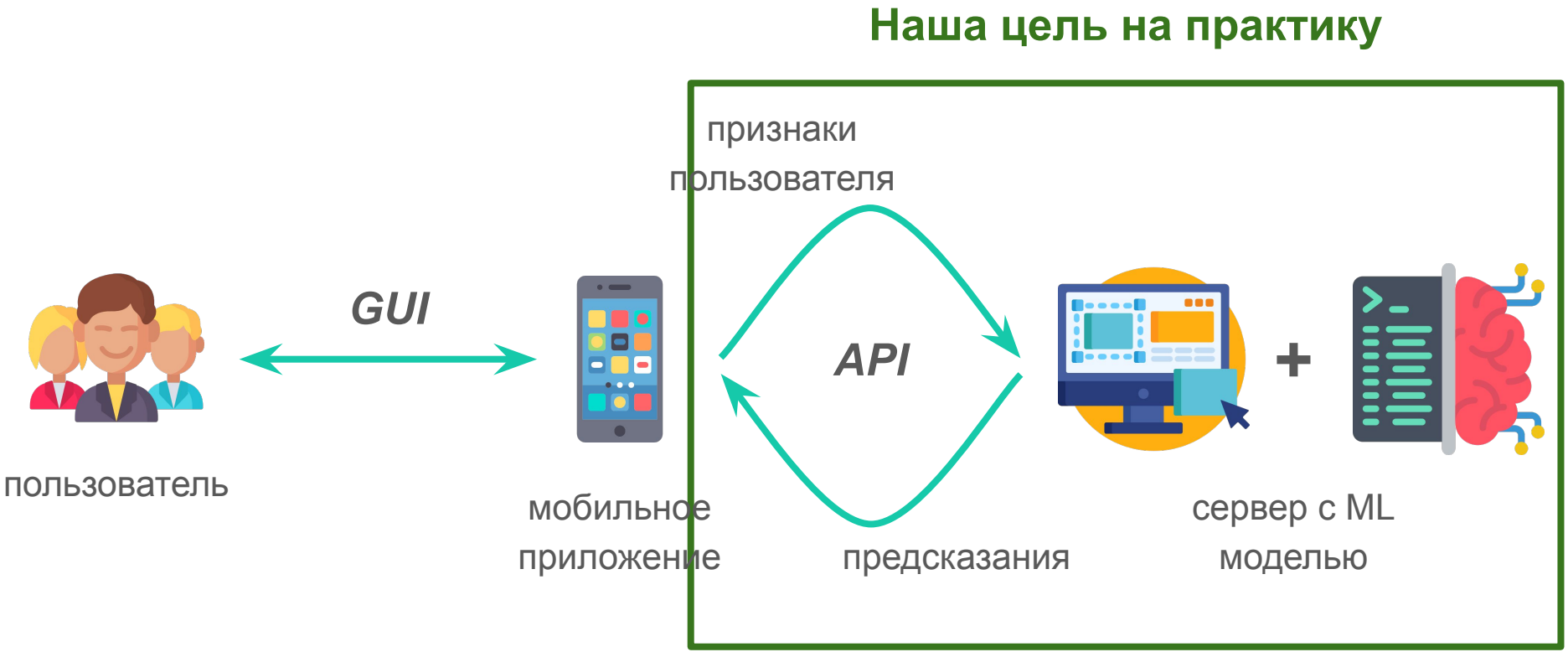
API **–** Application Programming Interface

GUI – Graphical User Interface

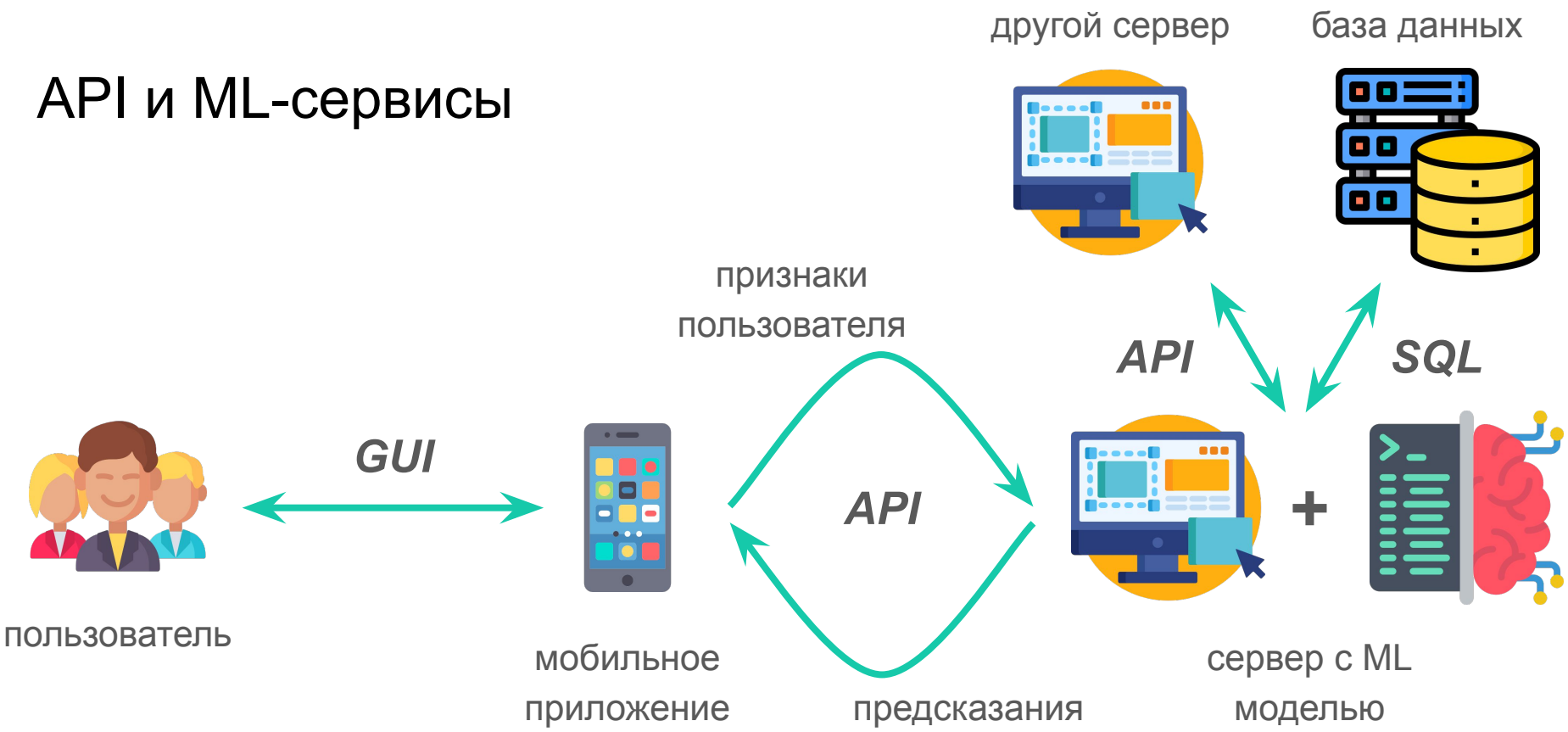
API и ML-сервисы



API и ML-сервисы



API и ML-сервисы



Типы запросов в API

- **GET** – получение ресурса
- **POST –** создание ресурса
- **PUT** – обновление ресурса
- **DELETE** – удаление ресурса

Никто не обязывает нас, как разработчиков сервера:

- Обрабатывать все 4 типа запросов
- Делать по запросу то действие, которому он отвечает

Библиотека FastAPI

- Открытый исходный код (open-source)
- Быстрая, с высокой производительностью
- Простой и интуитивный “питонячий” интерфейс
- Подробная документация, доступные туториалы
- Хорошее покрытие тестами, надежность

Сериализация и десериализация данных

Произвольная структура данных

- числа
- строки
- массивы
- словари
- картинки
- модели машинного обучения

сериализация

Последовательность бит

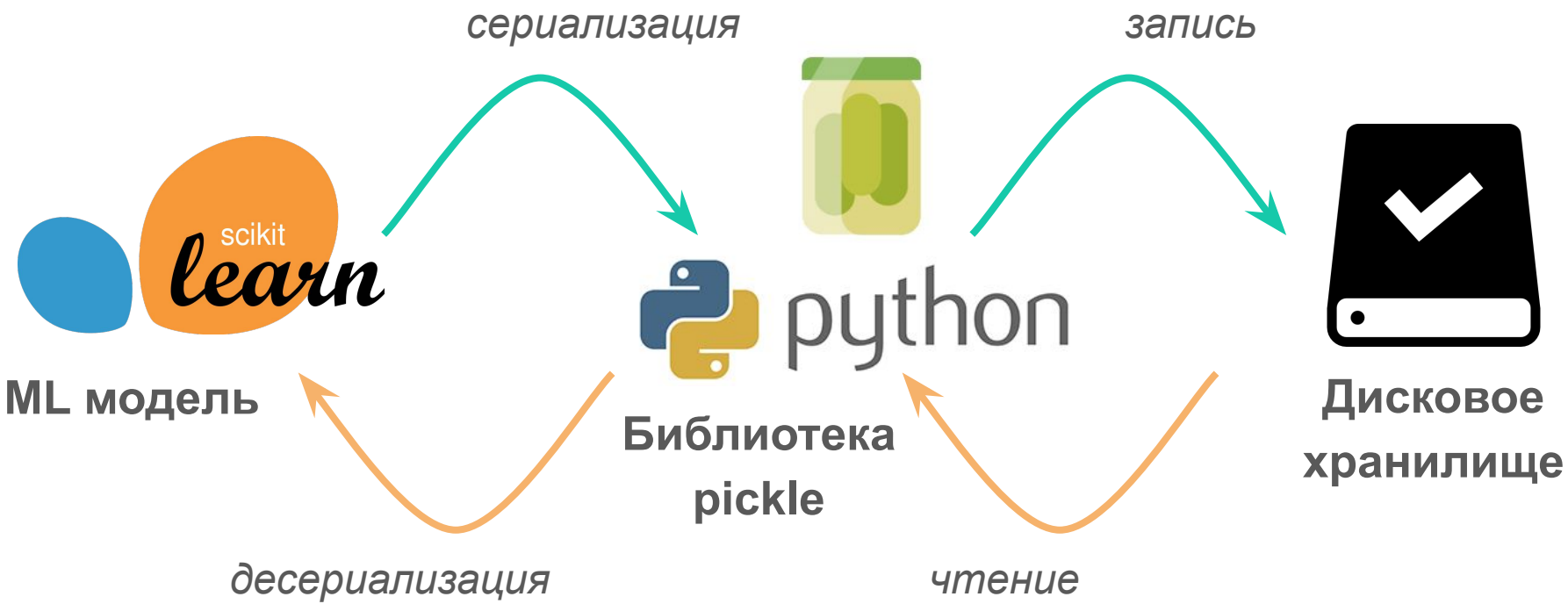
десериализация

Сериализация и десериализация данных

Зачем нужны?

- Сохранить структуру данных (обученную модель) на диск
- Передать на сервер, который хостит наш ML-сервис
- Загрузить модель и использовать ее для предсказаний

Сохранение и загрузка моделей из sklearn



Декораторы в Python

```python
def my_decorator(func):
    def wrapper():
        print('Message before function call')
        func()
        print('Message after function call')

    return wrapper


def say_hello():
    print('Hello, world!')


say_hello = my_decorator(say_hello)
```

```
In [3]: say_hello()
Message before function call
Hello, world!
Message after function call
```

Декораторы в Python

```python
def my_decorator(func):
    def wrapper():
        print('Message before function call')
        func()
        print('Message after function call')

    return wrapper


def say_hello():
    print('Hello, world!')


say_hello = my_decorator(say_hello)
```

```
In [3]: say_hello()
Message before function call
Hello, world!
Message after function call
```

```python
def my_decorator(func):
    def wrapper():
        print('Message before function call')
        func()
        print('Message after function call')

    return wrapper


@my_decorator
def say_hello():
    print('Hello, world!')
```

```
In [3]: say_hello()
Message before function call
Hello, world!
Message after function call
```

Вопросы?