



# API

SELEZNEV ARTEM  
SENIOR DE @ SBER

# НА СЕГОДНЯ

- REST или gRPC

{ REST }

↑ gRPC ↓

# НА СЕГОДНЯ

- REST или gRPC
- Python Framework

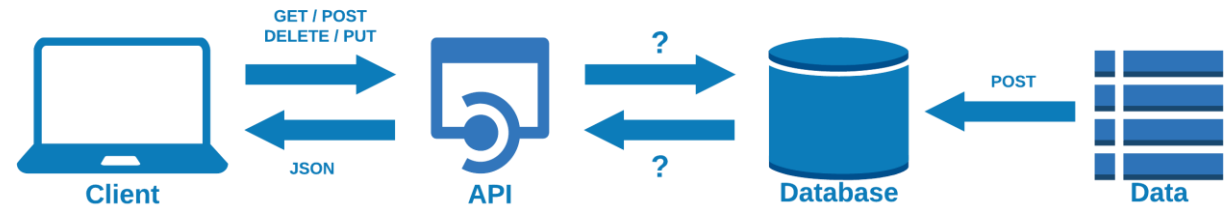


**django**



# НА СЕГОДНЯ

- REST или gRPC
- Python Framework
- FastAPI + Kedro



# НА СЕГОДНЯ

- REST или gRPC
- Python Framework
- FastAPI + Kedro
- BentoML



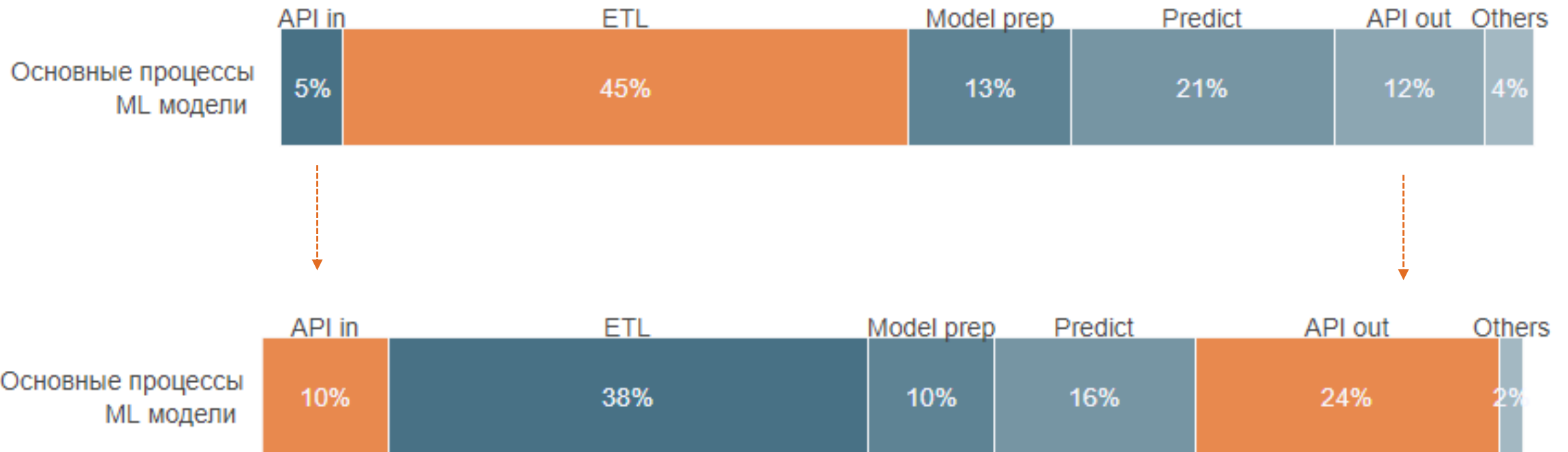
# ПРОБЛЕМЫ С ПРОБЛЕМАМИ



# API ПРОБЛЕМА



# API ПРОБЛЕМА





# API ПРОБЛЕМА | ЛОЖЬ ДАННЫХ



Источники: Национальный демографический отчет Департамент сельского хозяйства США

tylervigen.com

# API ПРОБЛЕМА | ЛОЖЬ ДАННЫХ

- `Int -> string`
- `decimal -> float`
- `string -> int`
- `categorical`


# API ПРОБЛЕМА | BIG 3

- External\_id
- Ответ != Запрос
- Зависимость от параметров  
 $\{\text{'a'} : [\{\text{'next'}: \text{'b'}\}]\} \rightarrow \{\text{'a'} : \{\text{'next'}: \text{'b'}\}\}$

# ПОЧЕМУ REST



?



Client sends a request

HTTP methods

Server sends a response

## REST

Интернет-протокол

Что такое REST?


Другие картинки

REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях это приводит к повышению производительности и упрощению архитектуры.

[Википедия](#)



?



# REST

Интернет-протокол

REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях это приводит к повышению производительности и упрощению архитектуры.

[Википедия](#)



# gRPC

A high-performance, open source universal RPC framework

# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0

# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML



# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML
Данные как есть	Данные бинаризованы

# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML
Данные как есть	Данные бинаризованы
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)

# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML
Данные как есть	Данные бинаризованы
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)
Медленный	Быстрый

# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML
Данные как есть	Данные бинаризованы
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)
Медленный	Быстрый



# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML
Данные как есть	Данные бинаризованы
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)
Медленный	Быстрый
Для людей	Для IoT

# СРАВНЕНИЕ

REST	gRPC
HTTP > 0	HTTP > 2.0
XML, JSON, CSV, RSS	XML
Данные как есть	Данные бинаризированы
Для всех ЯП	Только для ЯП с proto3* ( Java, C#, C++, Python, Ruby)
Медленный	Быстрый
Для людей	Для ИОТ

<https://github.com/grpc-ecosystem/awesome-grpc>

# СРАВНЕНИЕ

REST	gRPC	GraphQL
HTTP > 0	HTTP > 2.0	?
XML, JSON, CSV, RSS	XML	JSON
Данные как есть	Данные бинаризованы	Структурированные данные, по запросу
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)	Для всех ЯП (для кейсов, где важна типизация)
Медленный	Быстрый	Медленный
Для людей	Для ИОТ	Для людей (имеет типизацию)

<https://developer.github.com/v4/>

# СРАВНЕНИЕ

REST	gRPC	GraphQL
HTTP > 0	HTTP > 2.0	?
XML, JSON, CSV, RSS	XML	JSON
Данные как есть	Данные бинаризованы	Структурированные данные, по запросу
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)	Для всех ЯП (для кейсов, где важна типизация)
Медленный	Быстрый	Медленный
Для людей	Для IoT	Для людей (имеет типизацию)

“simultaneously sent too much data and didn’t include data that consumers needed”



# REST

REST	gRPC	GraphQL
HTTP > 0	HTTP > 2.0	?
XML, JSON, CSV, RSS	XML	JSON
Данные как есть	Данные бинаризованы	Структурированные данные, по запросу
Для всех ЯП	Только для ЯП с proto3* (Java, C#, C++, Python, Ruby)	Для всех ЯП (для кейсов, где важна типизация)
Медленный	Быстрый	Медленный
Для людей	Для ИОТ	Для людей (имеет типизацию)

БОЛЬШОЙ ВЫБОР



# DJANGO vs FLASK vs FASTAPI

```
* Serving Flask app "kedro_viz.server" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
```

# DJANGO vs FLASK vs FASTAPI

```
* Serving Flask app "kedro_viz.server" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
```

# DJANGO vs FLASK vs FASTAPI

WSGI  ASGI

# DJANGO vs FLASK vs FASTAPI



WSGI  ASGI

- Seriously impressive performance.
- WebSocket support.
- GraphQL support.
- In-process background tasks.
- Startup and shutdown events.
- CORS, GZip, Static Files, Streaming responses.
- Session and Cookie support.

# DJANGO vs FLASK vs FASTAPI

WSGI

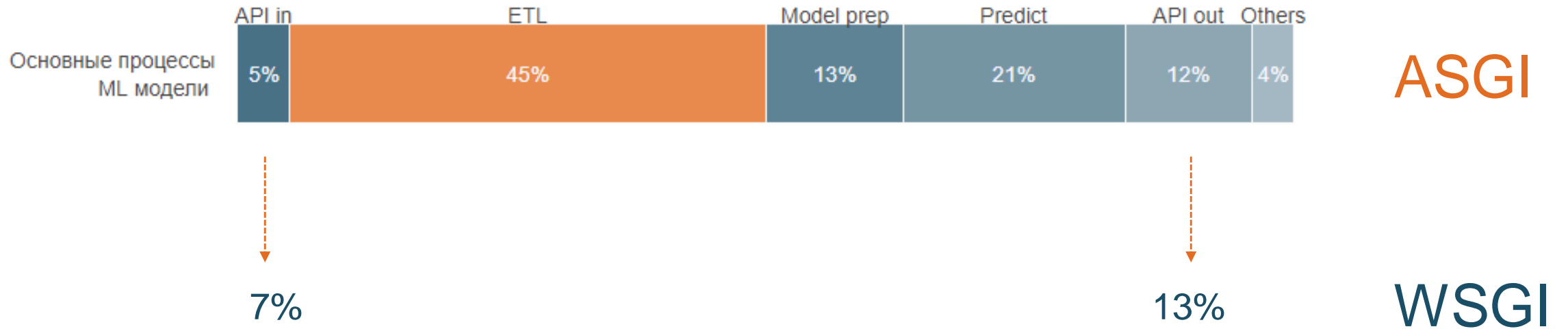


ASGI



· HTTP/2 and WebSockets

# DJANGO vs FLASK vs FASTAPI





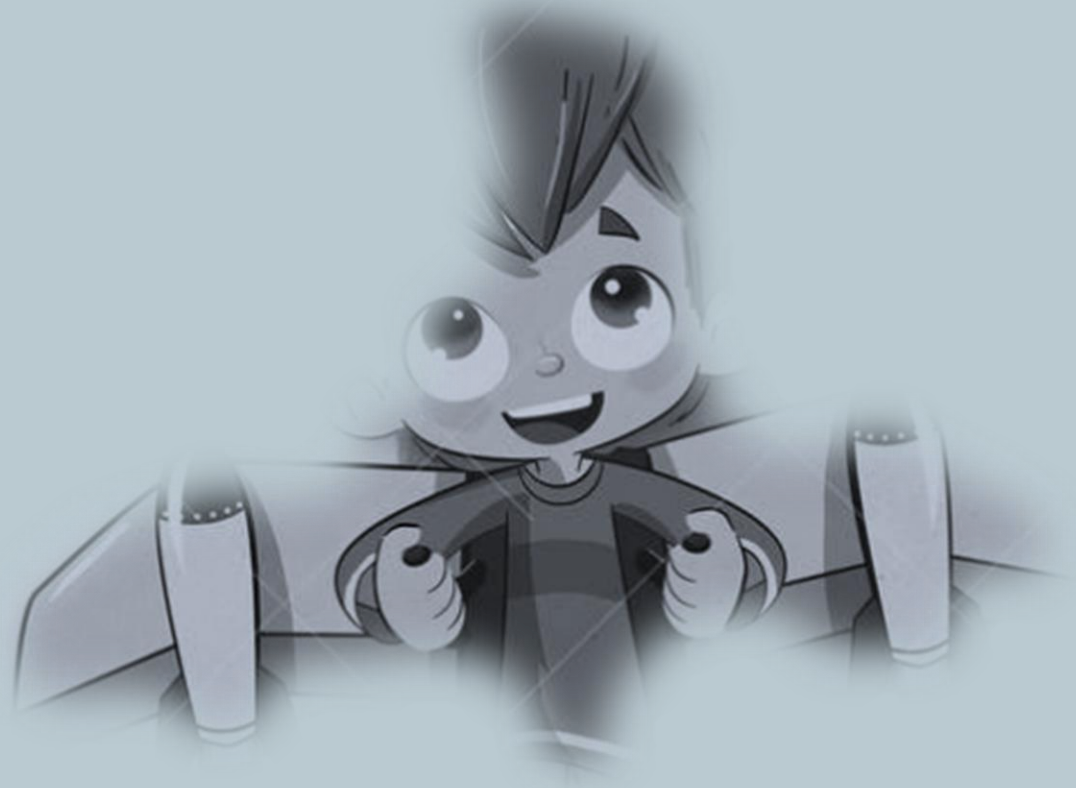
# FASTAPI + KEDRO



# ПАРУ СЛОВ О СУПЕР-КОРОБКЕ



ЗАДАНИЕ



# СОЗДАТЬ API

- Добавить в pipeline модели шаг, который будет работать с `APIDataSet`
- Добавленный pipeline должен только получать данные с API и делать предикт
- Написать файл, который будет содержать функцию подключения к добавленному pipeline и инициализацию его работы с данными
- Функция должна быть оформлена в FastAPI для отправки результатов предикта на адрес `localhost:port`
- За основу «сервера данных» можно взять пример из практической части презентации