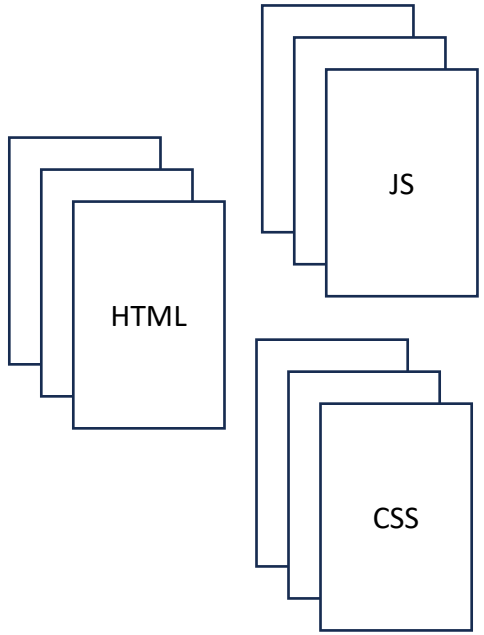
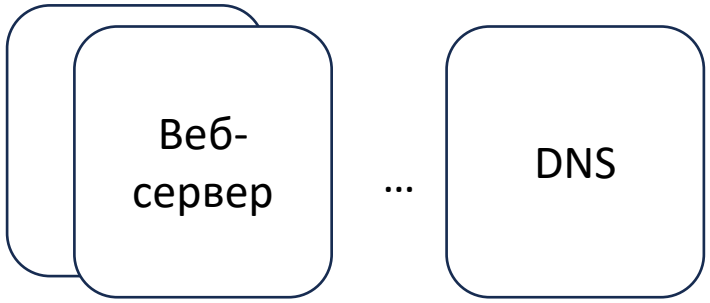
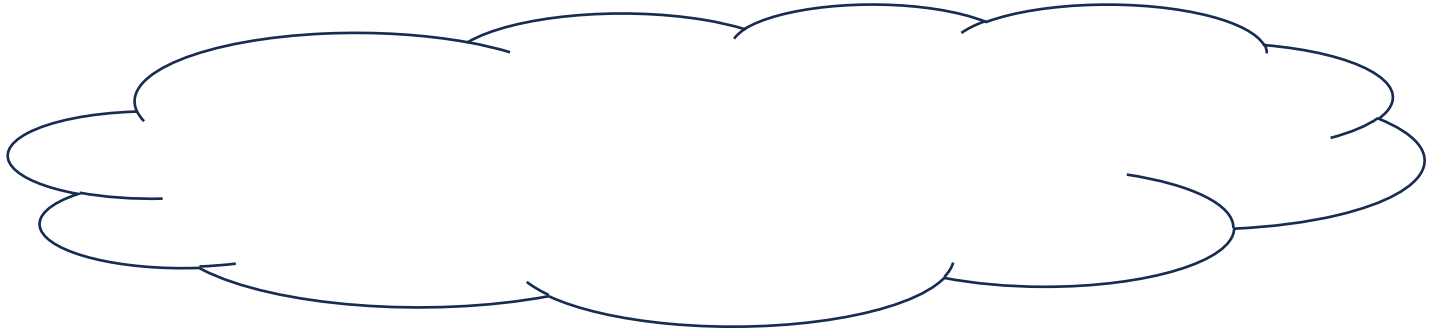
Архитектура веб-приложений

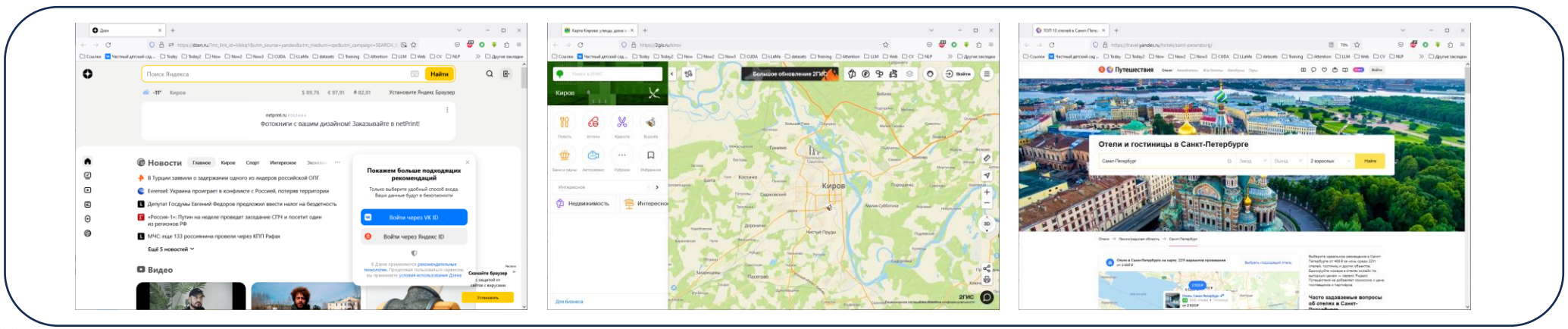
Татаринова А.Г., каф. ПМИ

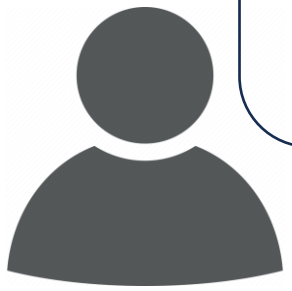
2025


HTML
JS
CSS

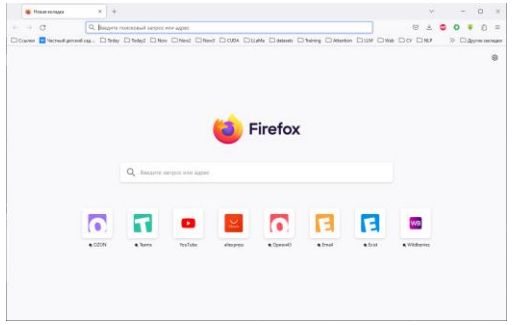

Веб-
сервер
...
DNS




Найти
Покажем больше подходящих рекомендаций
Войти через VK ID
Войти через Яндекс ID
Новости
Видео
Киров
Недвижимость
Отели и гостиницы в Санкт-Петербурге



Получение страницы в браузере



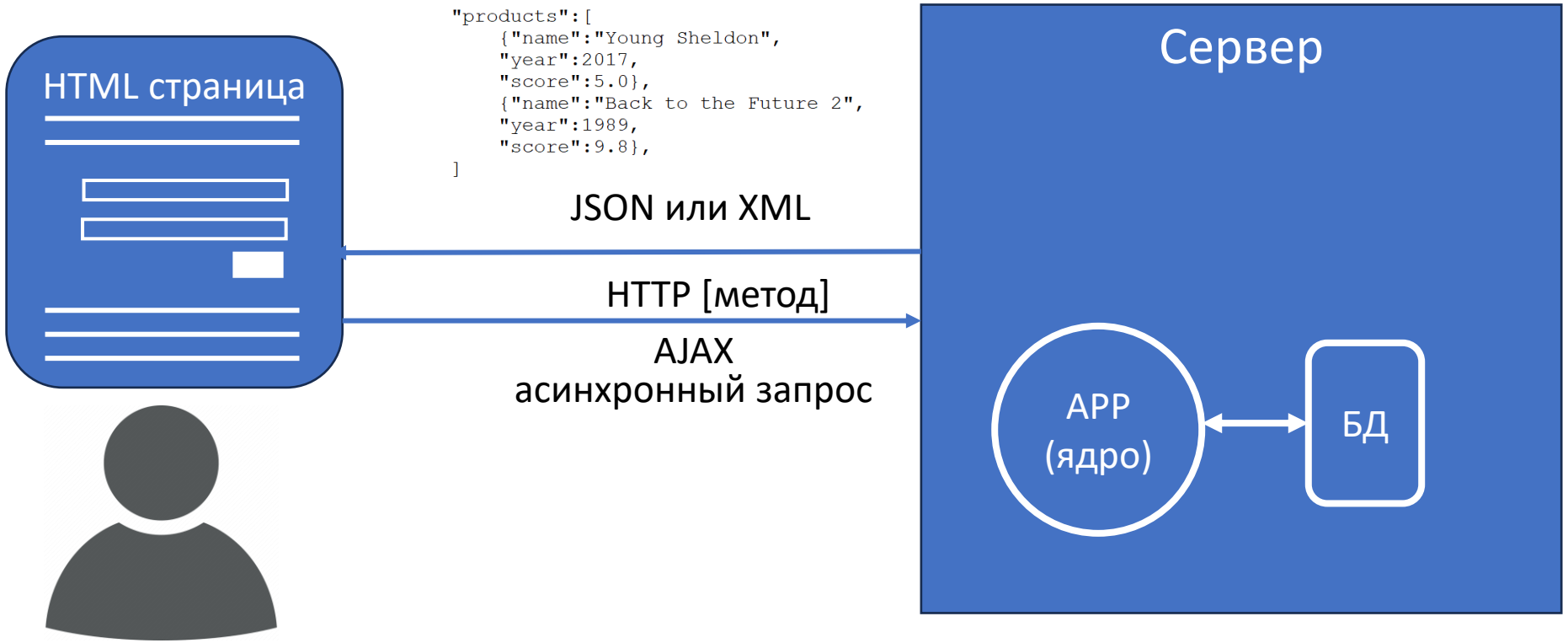
HTML страница

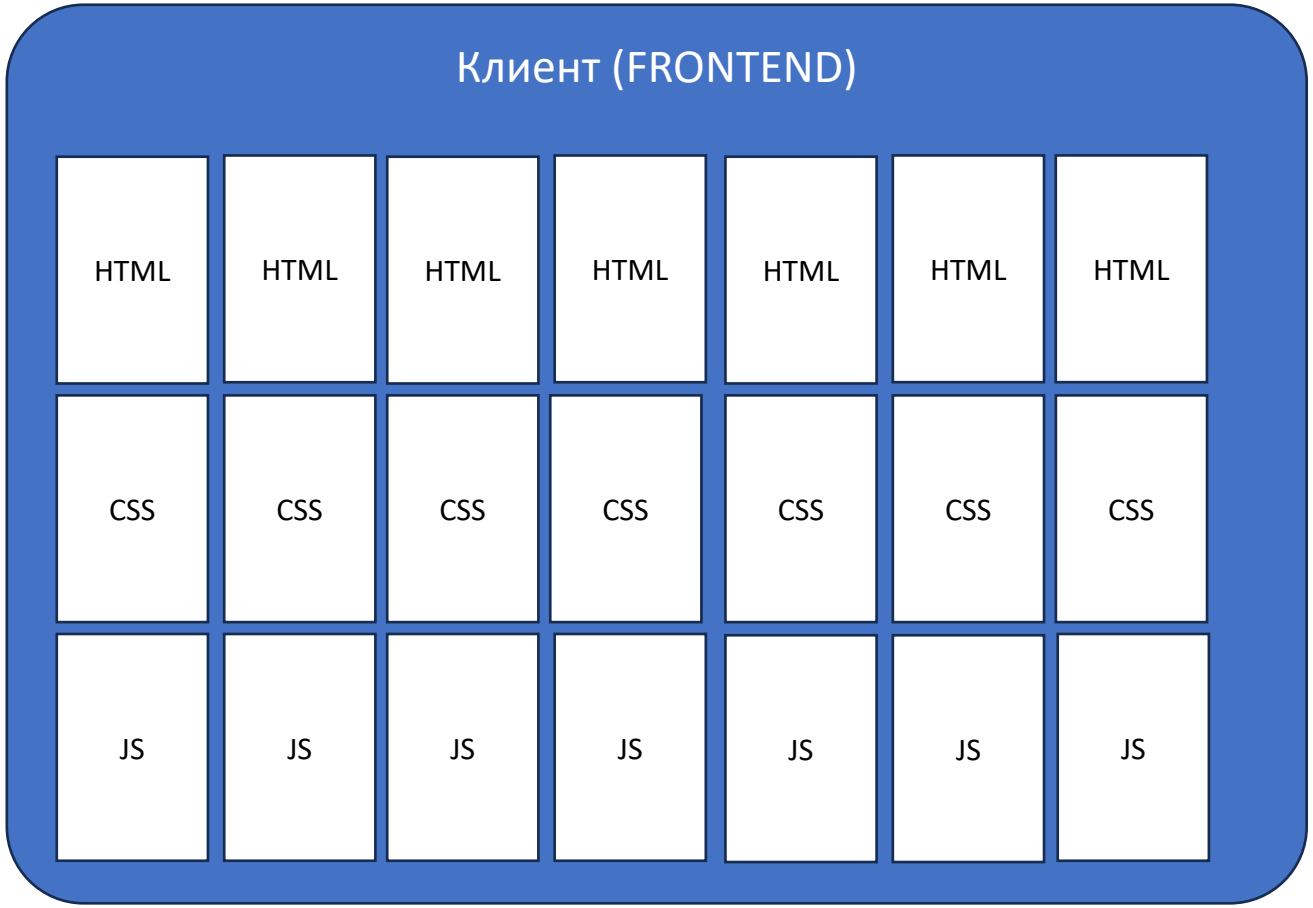
HTTP GET

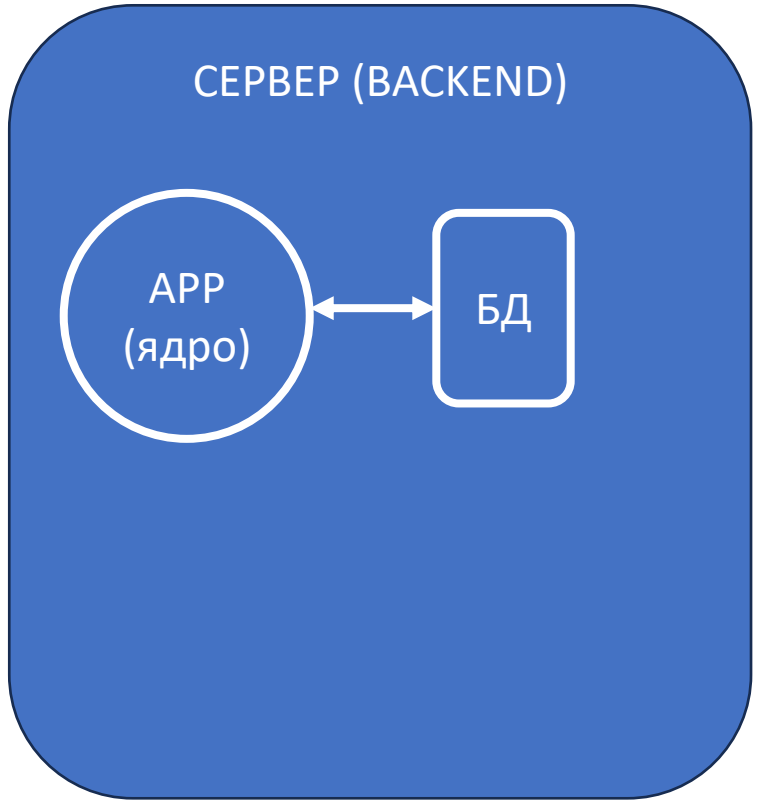
Сервер

- слушает запросы
- отдает статические веб-страницы

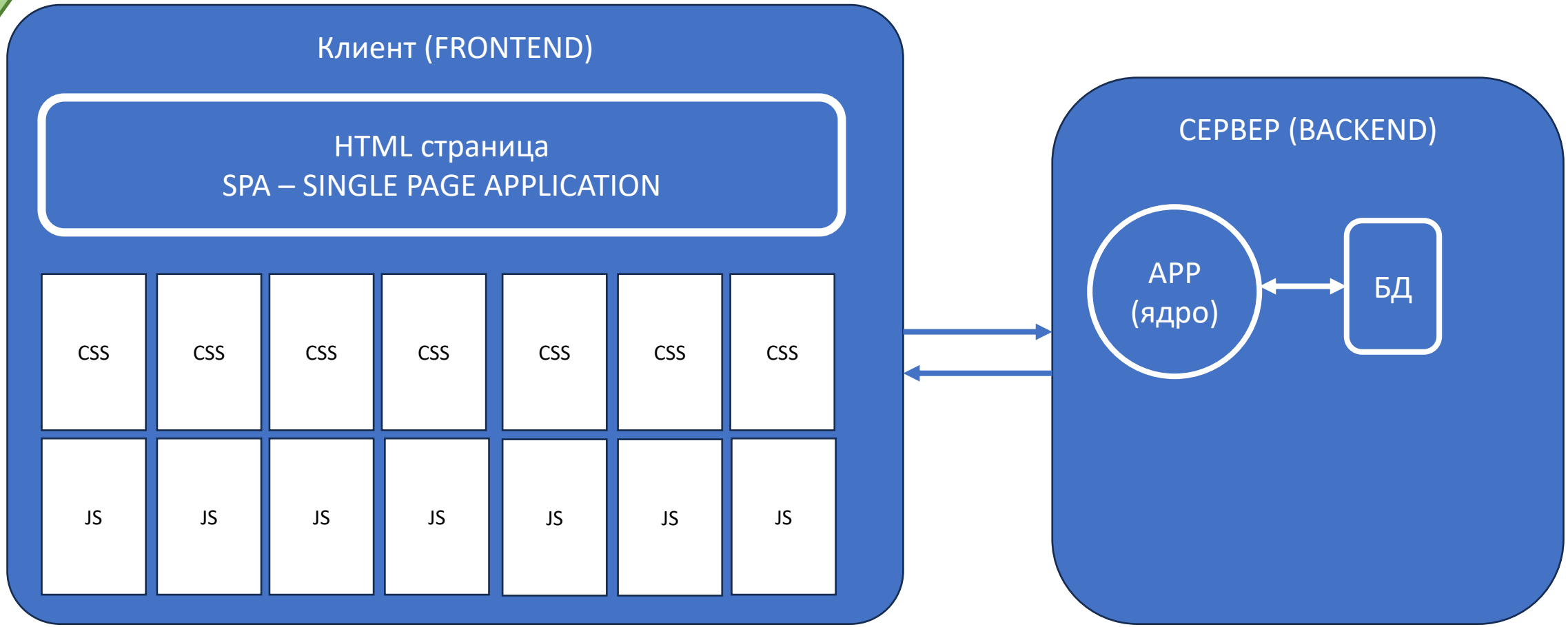
Получение страницы в браузере



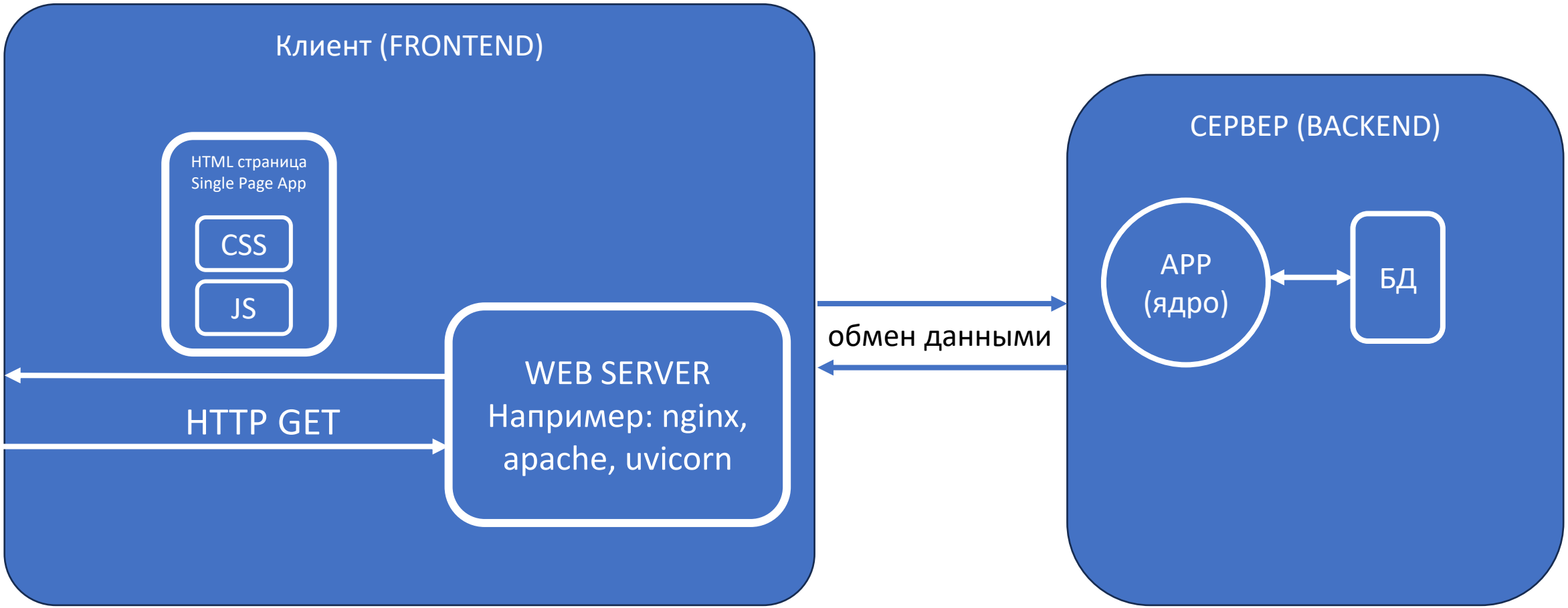

Клиент (FRONTEND)
HTML
HTML
HTML
HTML
HTML
HTML
HTML
CSS
CSS
CSS
CSS
CSS
CSS
CSS
JS
JS
JS
JS
JS
JS
JS

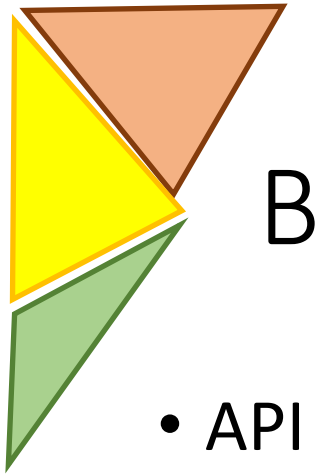

СЕРВЕР (BACKEND)
APP
(ядро)
БД

SPA


Клиент (FRONTEND)
HTML страница
SPA – SINGLE PAGE APPLICATION
CSS
CSS
CSS
CSS
CSS
CSS
CSS
JS
JS
JS
JS
JS
JS
JS
СЕРВЕР (BACKEND)
APP
(ядро)
БД

Frontend vs Backend

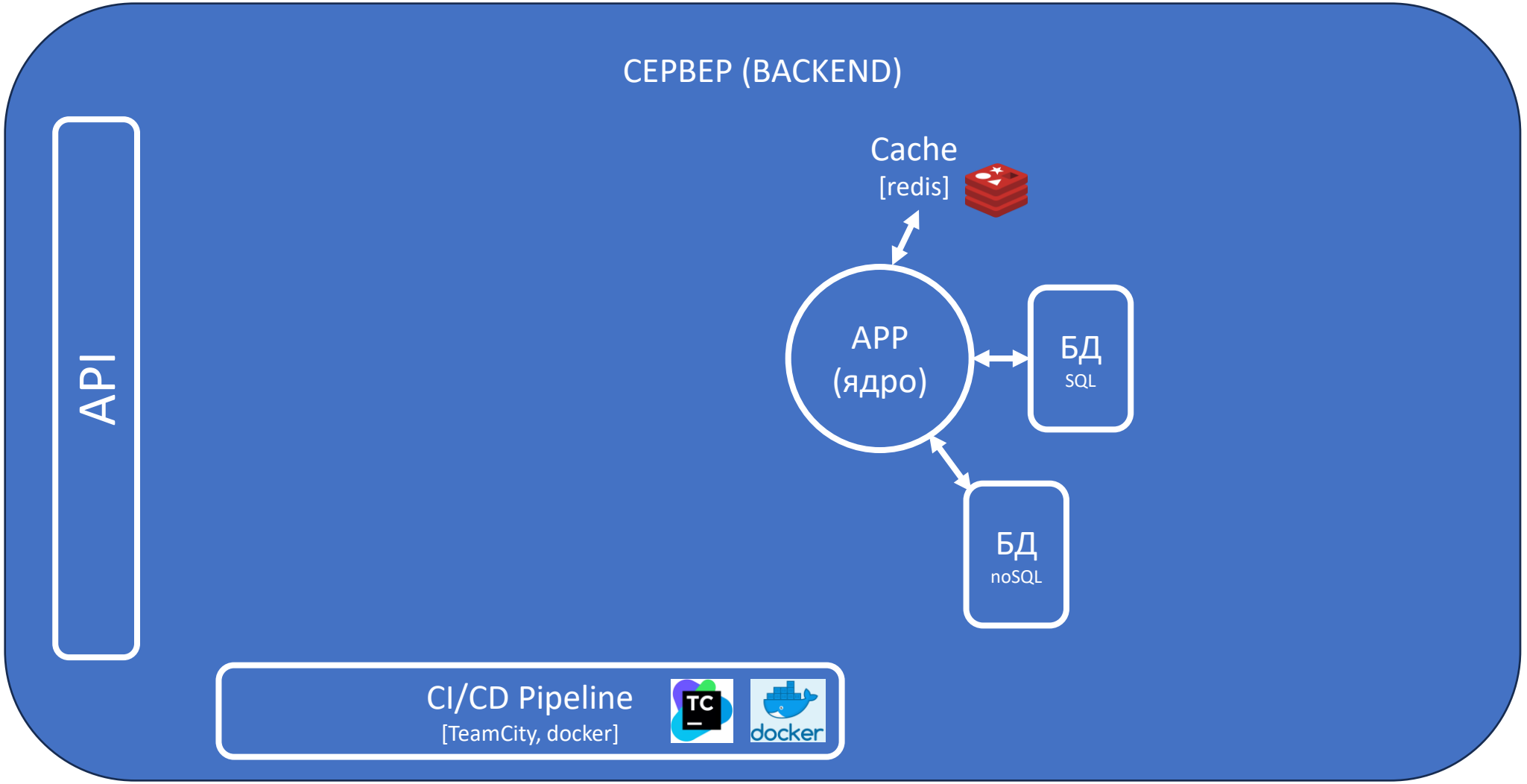




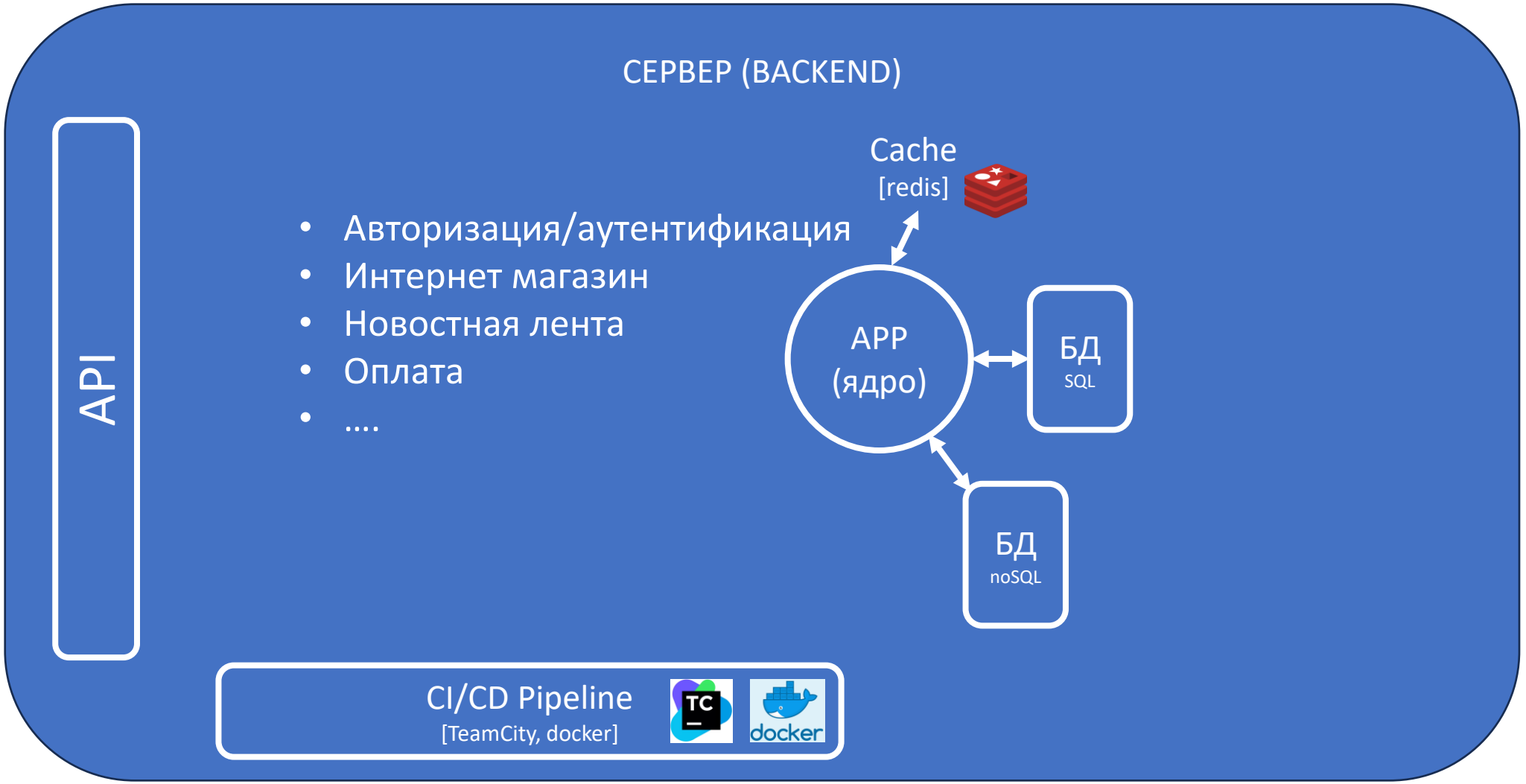
Backend

- API (Application Programming Interfaces)
- Серверная сторона веб-API состоит из одного или нескольких публично доступных конечных точек для системы пересылки сообщений запрос-ответ, обычно выраженных в формате JSON или XML, передаваемых с помощью веб-сервера на основе HTTP

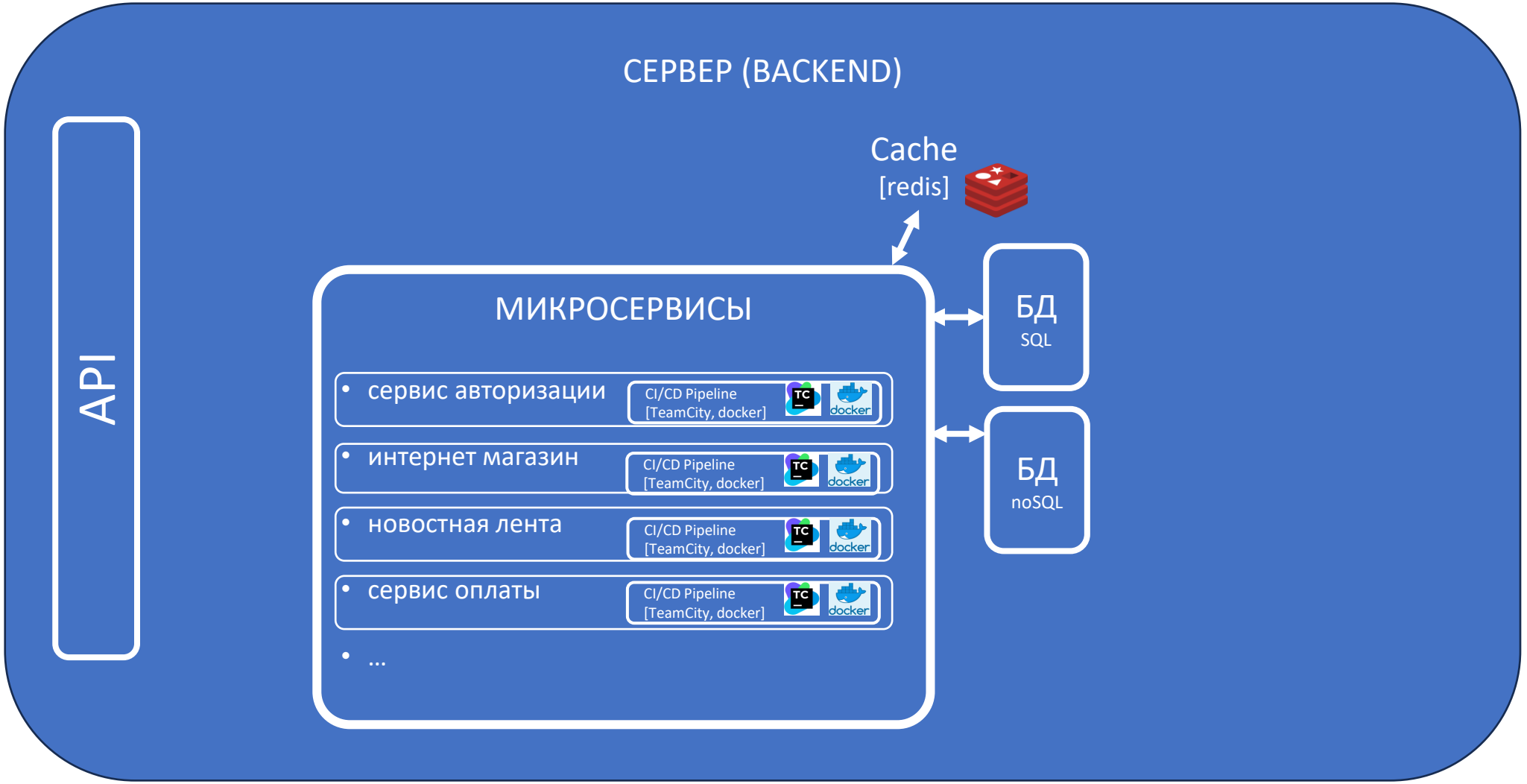
Backend


СЕРВЕР (BACKEND)
API
Cache
[redis]
APP
(ядро)
БД
SQL
БД
noSQL
CI/CD Pipeline
[TeamCity, docker]
TC
docker

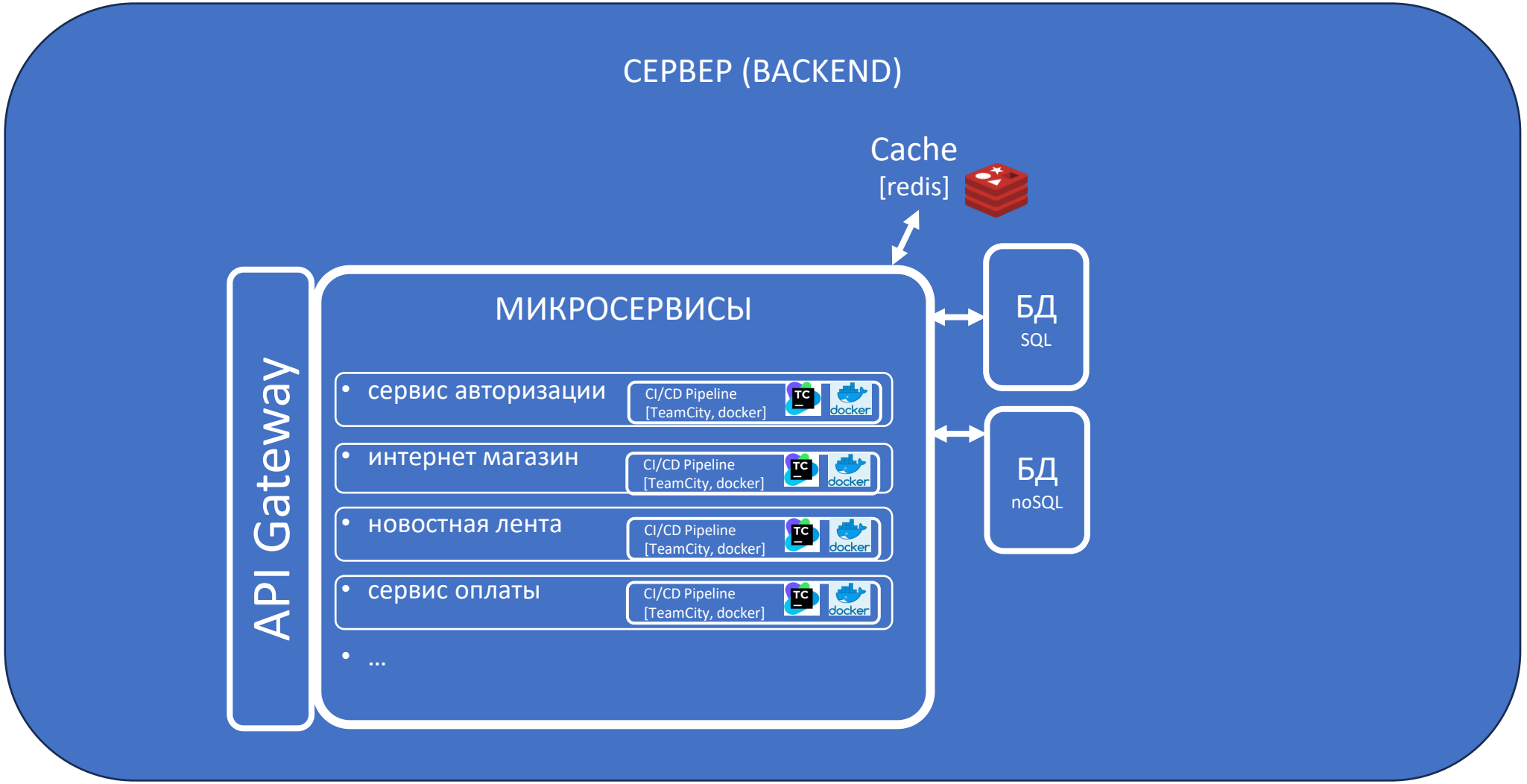
Backend


СЕРВЕР (BACKEND)
API
Авторизация/аутентификация
Интернет магазин
Новостная лента
Оплата
….
Cache
[redis]
APP
(ядро)
БД
SQL
БД
noSQL
CI/CD Pipeline
[TeamCity, docker]
TC
docker

Backend

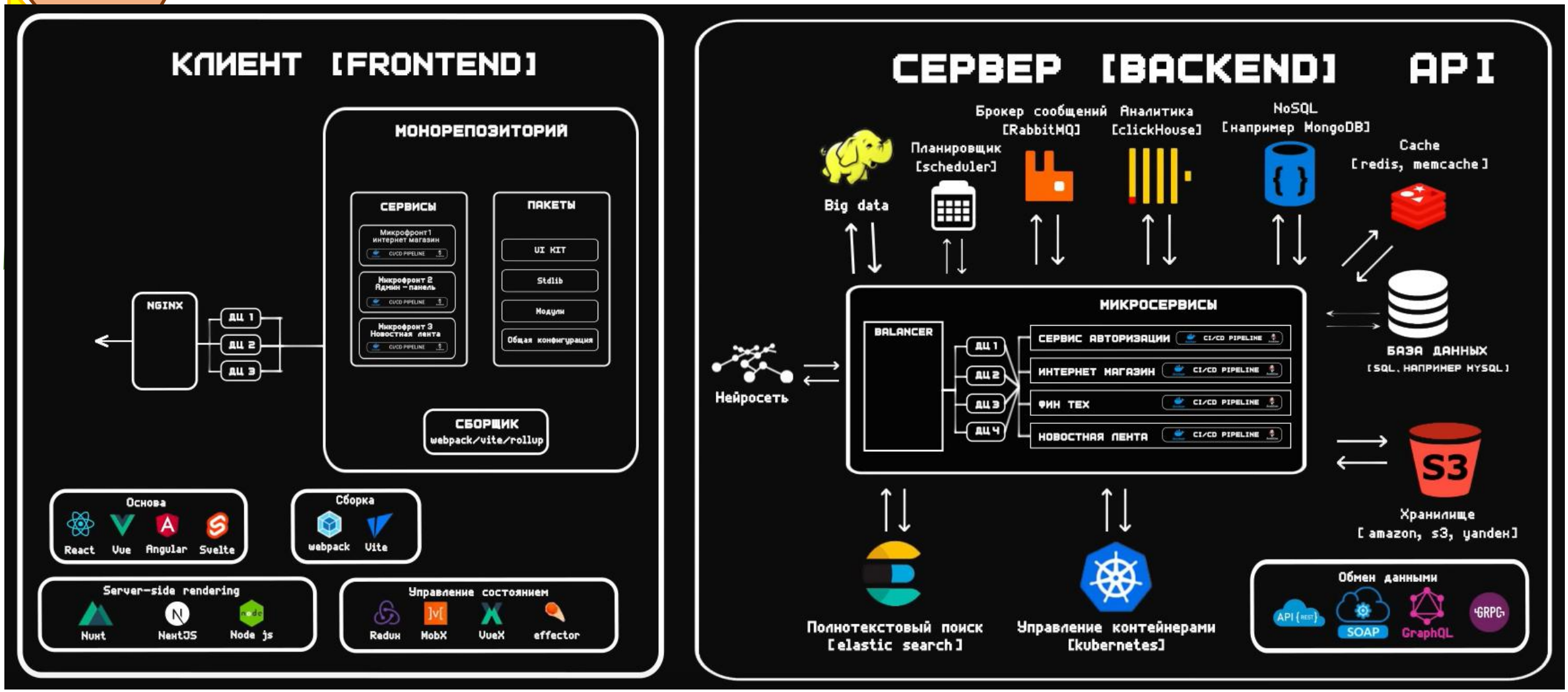

СЕРВЕР (BACKEND)
API
Cache
[redis]
МИКРОСЕРВИСЫ
• сервис авторизации
CI/CD Pipeline
[TeamCity, docker]
• интернет магазин
CI/CD Pipeline
[TeamCity, docker]
• новостная лента
CI/CD Pipeline
[TeamCity, docker]
• сервис оплаты
CI/CD Pipeline
[TeamCity, docker]
• ...
БД
SQL
БД
noSQL

Backend

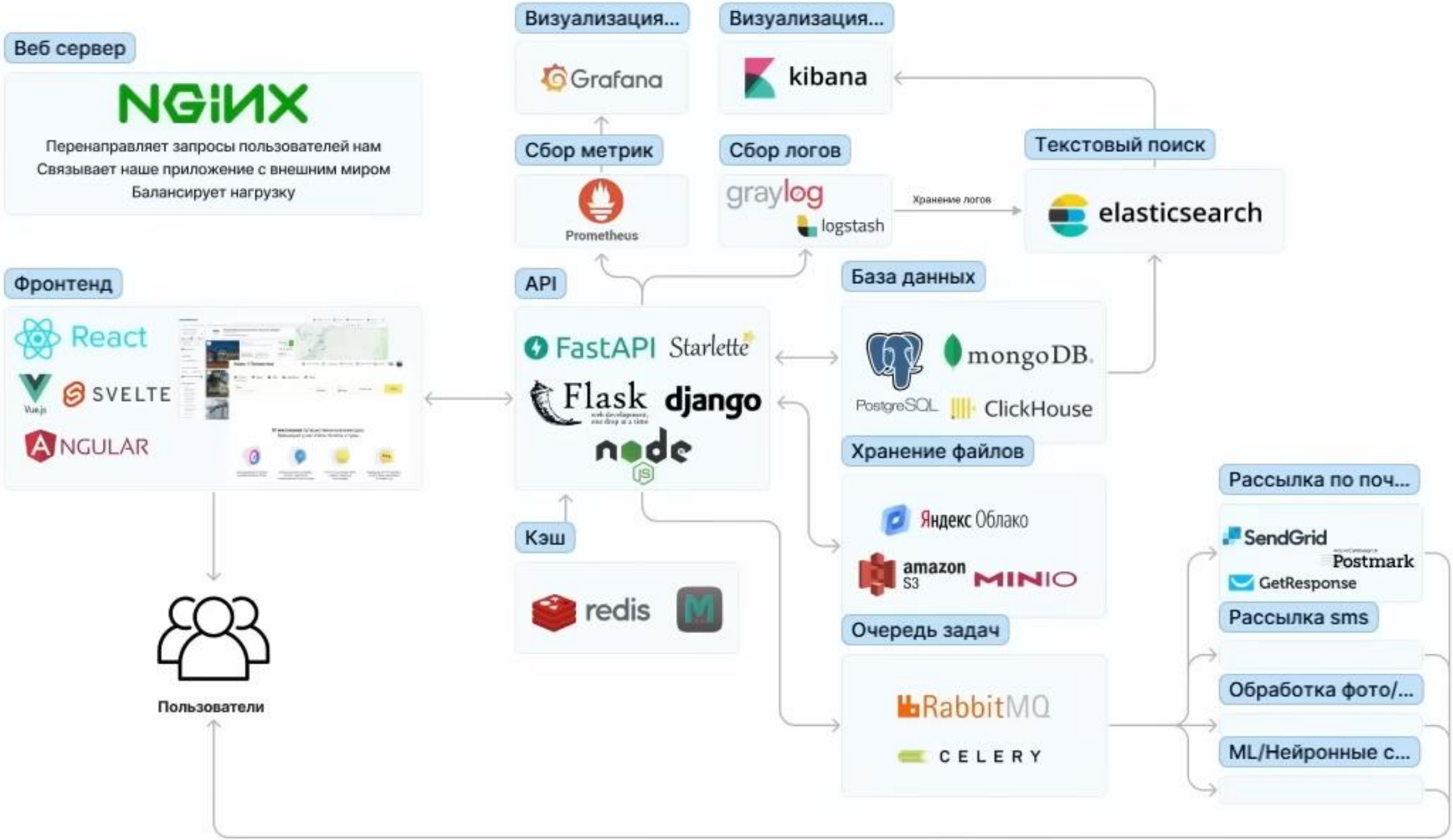

СЕРВЕР (BACKEND)
Cache
[redis]
API Gateway
МИКРОСЕРВИСЫ
• сервис авторизации
CI/CD Pipeline
[TeamCity, docker]
• интернет магазин
CI/CD Pipeline
[TeamCity, docker]
• новостная лента
CI/CD Pipeline
[TeamCity, docker]
• сервис оплаты
CI/CD Pipeline
[TeamCity, docker]
• ...
БД
SQL
БД
noSQL

Backend

- API-шлюз (API Gateway) — программное обеспечение, которое принимает запрос пользователя приложения, перенаправляет его к одному или нескольким бэкенд-сервисам, собирает необходимые данные и возвращает их пользователю в виде единого, объединённого пакета.
  Кроме того, он обеспечивает аналитику, уровни защиты от угроз и другие меры безопасности для приложения.



* - https://www.youtube.com/watch?v=S0e_5a2WB60&list=WL&index=89



* - https://www.youtube.com/watch?v=ZgojwJEcLn8&list=WL&index=90

FastAPI

- Фреймворк для создания быстрых HTTP API-серверов со встроенными валидацией, сериализацией и асинхронностью
- Использует два других фреймворка:
  - Starlette – работа с web,
  - Pydantic – отвечает за валидацию
- Для работы FastAPI необходим ASGI-сервер, по-умолчанию документация предлагает uvicorn
- Примеры: микросервисы Uber, Microsoft, Netflix
  vllm, tools для LLM, RAG-endpoints

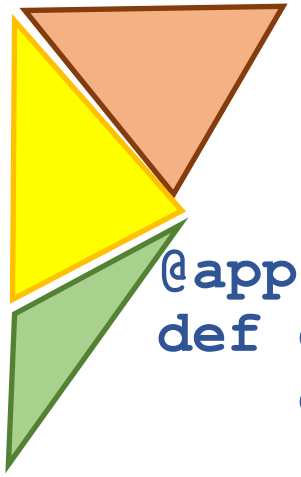
- Общая схема URL
  <scheme>://<host>:<port>/<path>?<query>#<fragment>
- Пример URL
  https://www.example.com:8080/folder/page.html?id=123&name=example#section1

- Обработка различных типов запросов HTTP
- В классе FastAPI для каждого из типов запросов определены одноименные методы, которые применяются в качестве декоратора к функциям, непосредственно обрабатывающие запрос
- Пример:

```
from fastapi import FastAPI
app = FastAPI()

@app.get("/")
def root():
    return {"message": "Hello world"}

@app.get("/example_0")
def index():
    return {"message": “Example"}
```

- Отправка ответов
- Класс fastapi.Response является базовым для остальных классов ответа (по умолчанию JSONResponse)
- Пример:

```
@app.get("/example_1")
def get_1():
    return JSONResponse(content={"message": "Hello world"})

@app.get("/example_2")
def get_2():
    data = "Hello"
    return Response(content=data, media_type="text/plain")
```



```
@app.get("/example_3")
def get_3():
    data = "<h2>Hello world</h2>"
    return HTMLResponse(content=data)

@app.get("/example_4")
def get_4():
    data = "Hello world"
    return Response(content=data, media_type="text/plain")

@app.get("/example_5", response_class = HTMLResponse)
def get_5():
    return "<h2>Hello world</h2>"
```

- Путь запроса может содержать специальные значения, которые фреймворк FastAPI может связать с параметрами функции обработчика запроса
- Класс **Path** позволяет наложить ограничения на значения параметров
- Пример:

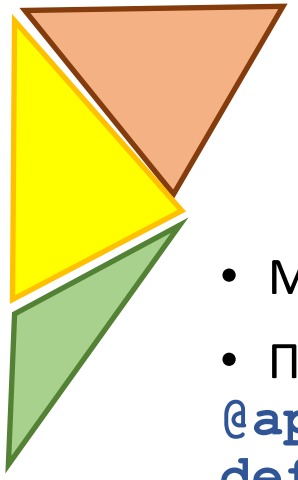
```
@app.get("/users/{id}")
def users_1(id):
    return {"user_id": id}

@app.get("/users/{name}-{age}")
def users_2(name, age):
    return {"user_name": name, "user_age": age}

@app.get("/users/{name}")
def users_3(name:str = Path(min_length=3, max_length=20)):
    return {"name": name}
```

- Для получения значений параметров строки запроса можно в функции определить одноименные параметры
- Пример для http://127.0.0.1:8000/users/add?name=Tom&age=38 :

```
@app.get("/users")
def get_model(name, age):
    return {"user_name": name, "user_age": age}

@app.get("/users")
def get_model(name = "Undefined", age = 18):
    return {"user_name": name, "user_age": age}

@app.get("/users")
def get_model(name: str, age: int = 18):
    return {"user_name": name, "user_age": age}
```

- Дополнительно для работы с параметрами строки запроса фреймворк предоставляет класс **Query**
- Пример:

```
@app.get("/users")
def users(name:str  = Query(min_length=3, max_length=20)):
    return {"name": name}

@app.get("/users")
def users(phone:str  = Query(pattern=r"^\d{11}$")):
    return {"phone": phone}

@app.get("/users")
def users(name: str = Query(default="Undefined", min_length=2)):
    return {"name": name}

@app.get("/users")
def users(name:str | None = Query(default=None, min_length=2)):
    if name==None:
        return {"name": "Undefined"}
    else:
        return {"name": name}
```



- Можно получать списки значений
- Пример для http://127.0.0.1:8000/users?people=tom&people=Sam&people=Bob:

```
@app.get("/users")
def users(people: list[str]  = Query()):
    return {"people": people}
```

- Пример сочетание параметров пути и запроса:

```
@app.get("/users/{name}")
def users(name:str  = Path(min_length=3, max_length=20),
          age: int = Query(ge=18, lt=111)):
    return {"name": name, "age": age}
```

- Отправка статусных кодов, которые указывают на статус выполнения операции на сервере

```
@app.get("/notfound", status_code=404)
def notfound():
    return  {"message": "Resource Not Found"}

@app.get("/users/{id}", status_code=200)
def users(response: Response, id: int = Path()):
    if id < 1:
        response.status_code = 400
        return {"message": "Incorrect Data"}
    return  {"message": f"Id = {id}"}
```