



Объектно-ориентированное программирование

2023

План лекции

Рассмотрим пример создания простого серверного приложения с использованием C++.

1. Постановка задачи
2. Выбор технологий
 - Poco library
 - MariaDB
 - Docker
3. Простое web-приложение
4. Добавляем REST интерфейс
5. Добавляем базу данных
6. Заполняем базу тестовыми данными
7. Проводим нагружочное тестирование

Постановка задачи

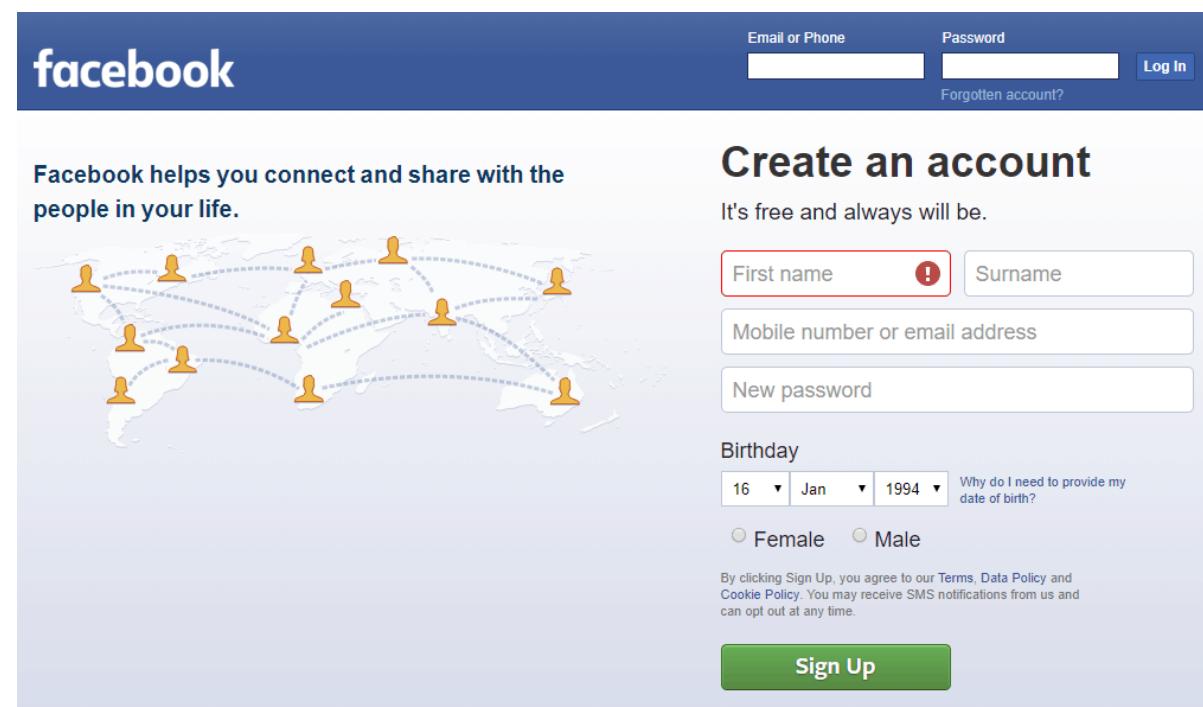
В большинстве веб-приложений так или иначе хранится информация о пользователях/клиентах. Давайте сделаем небольшое веб-приложение, которое позволяет хранить информацию о пользователях, добавлять ее и искать.

О пользователях на надо знать:

- имя
- фамилию
- e-mail
- как обращаться
- логин/пароль

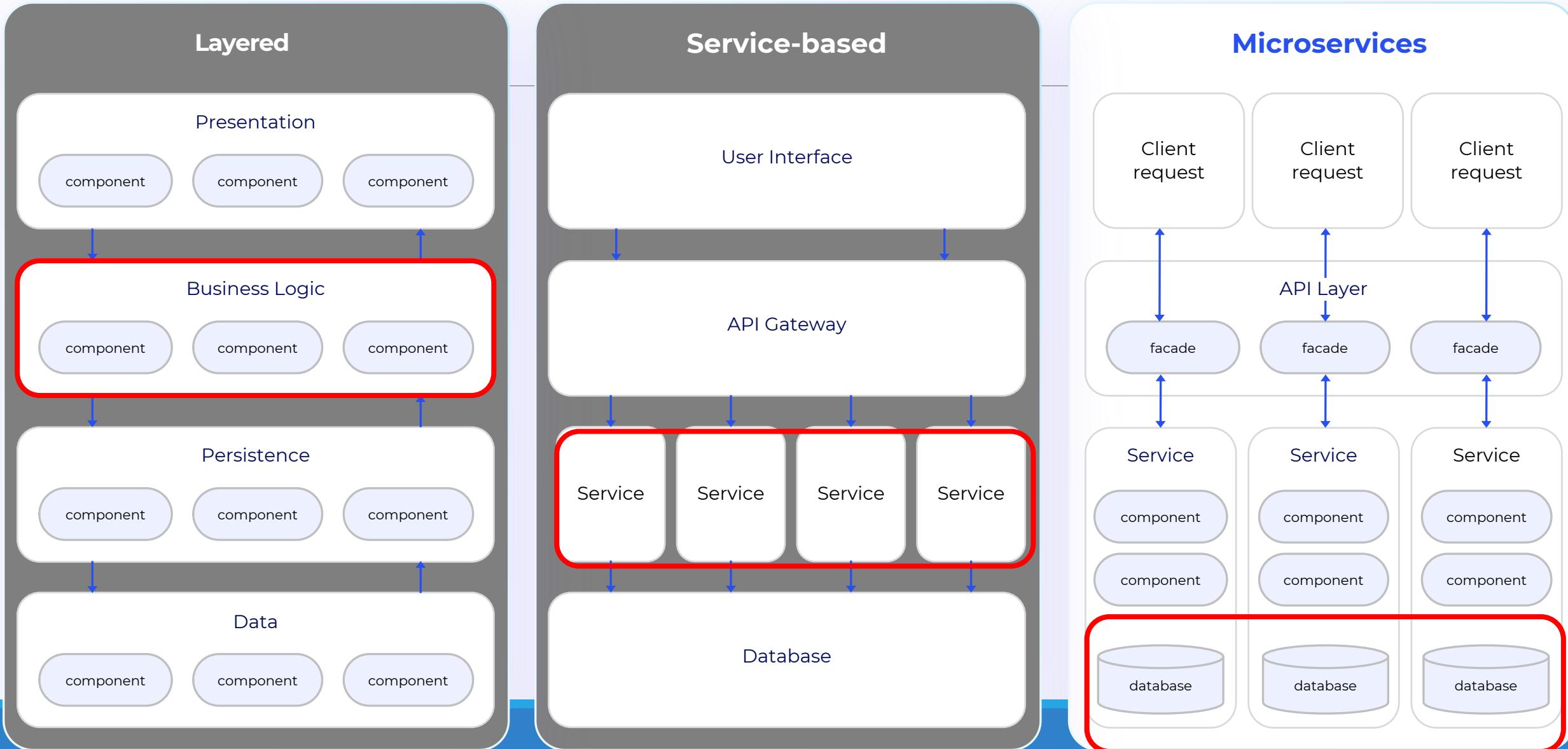
Искать пользователей будем:

- по id
- по маске имени и фамилии



The screenshot shows the Facebook sign-up page. At the top, there's a blue header with the word "facebook". Below it, there are fields for "Email or Phone" and "Password", along with a "Log In" button and a "Forgotten account?" link. The main area features a world map with orange user icons and dashed lines connecting them, symbolizing a global network. The text "Facebook helps you connect and share with the people in your life." is displayed above the map. To the right, there's a large green "Create an account" button. Below it, the text "It's free and always will be." is followed by input fields for "First name" (with a red info icon), "Surname", "Mobile number or email address", and "New password". There are dropdown menus for "Birthday" (set to 16 Jan 1994) and gender selection ("Female" and "Male" radio buttons). At the bottom, there's a link "Why do I need to provide my date of birth?", a "Sign Up" button, and a small note about agreeing to terms and policies.

Архитектурный стиль





Выбор
технологий

Сервер = linux

- Если вы используете Windows, то убедитесь что на нем установлен WSL2 (Windows Subsystem for Linux)
<https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-11-with-gui-support>
- Если вы используете Mac – то понадобится среда виртуализации:
 - для Intel x86 - <https://www.virtualbox.org/wiki/Downloads>
 - для M1/2 <https://mac.getutm.app/>

В качестве базового образа используем:

<https://ubuntu.com/download/server> - для x86

<https://ubuntu.com/download/server/arm> - для Mac M1/M2



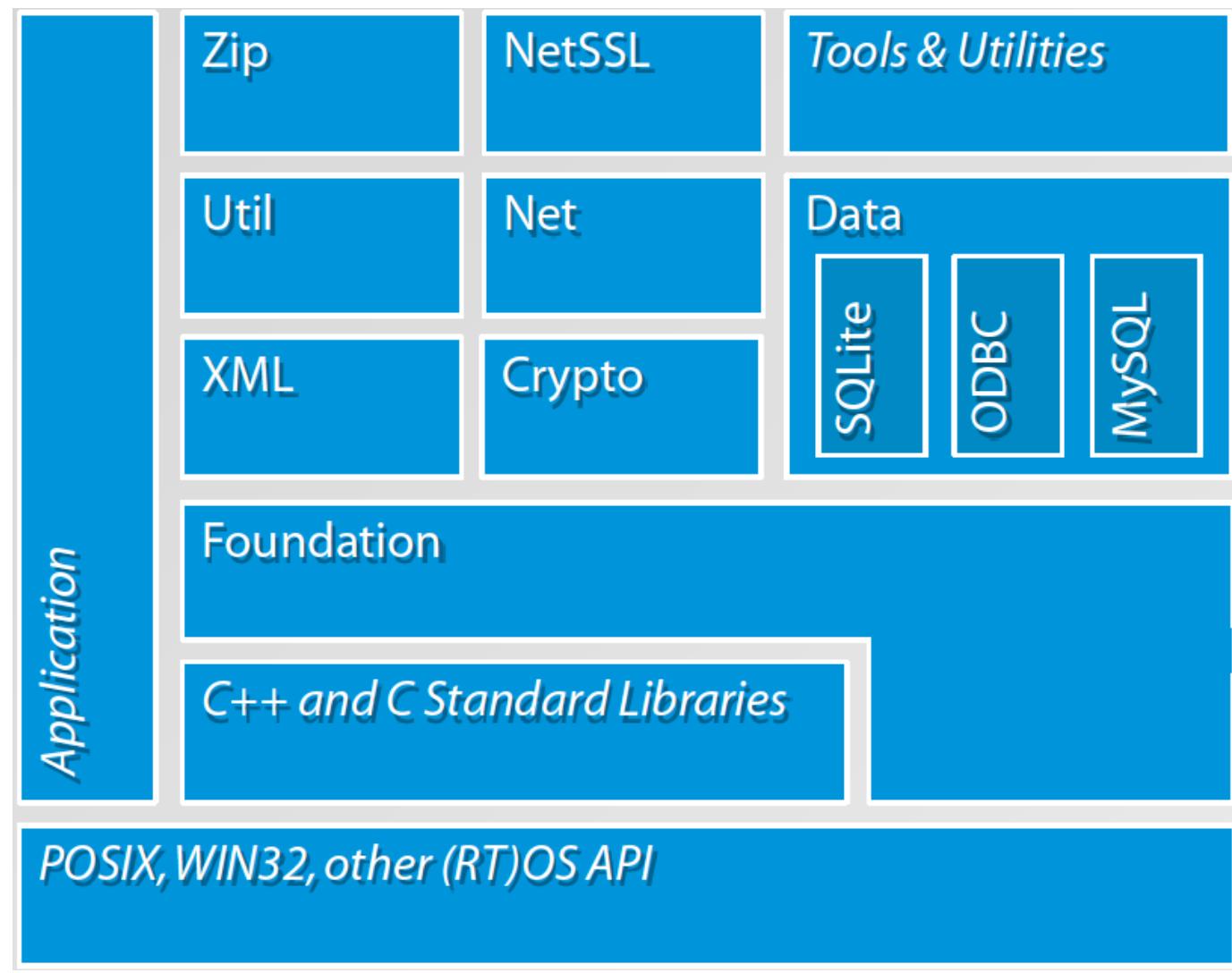
Cross-platform open source C++ libraries
that simplify desktop, server, mobile, IoT
and embedded software development.

<https://pocoproject.org>

роCO

- Коллекция C++ библиотек;
- Сфокусирована на создание сетевых приложений и микросервисов;
- Использует современный ANSI/ISO Standard C++ и базируется на C++ Standard Library/STL;
- Обладает высокой степенью портируемости;
- Легковесна и подходит для IoT устройств;
- Открытая лицензия на основе Boost Software License,
- Можно использовать в коммерческих и не коммерческих проектах;

Структура



external dependencies

- Базовый вариант (Foundation, XML, Util, Net) не требует зависимостей;
- Полное решение зависит от:
 - OpenSSL (Crypto and NetSSL)
 - ODBC (Data/ODBC)
 - MySQL Client (Data/MySQL)

Установка

```
# Устанавливаем необходимые пакеты
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install -y build-essential autoconf libtool pkg-
config gcc g++ cmake git libssl-dev zlib1g-dev mysql-client
libmysqlclient-dev libboost-all-dev docker docker-compose

# Устанавливаем библиотеку Poco
git clone -b poco-1.12.4-release
https://github.com/pocoproject/poco.git
cd poco
mkdir cmake-build
cd cmake-build
cmake ..
cmake --build . --config Release
sudo cmake --build . --target install
cd .
```

Как подключить библиотеку?

```
cmake_minimum_required(VERSION 3.2)

find_package(Threads)
find_package(OpenSSL)
find_package(Poco REQUIRED COMPONENTS Foundation Util Net XML
JSON Crypto NetSSL)

if(NOT ${Poco_FOUND})
  message(FATAL_ERROR "Poco C++ Libraries not found.")
endif()

include_directories(${Poco_INCLUDE_DIRS})

add_executable(poco_thread poco_thread.cpp)

target_link_libraries(poco_thread PRIVATE
${CMAKE_THREAD_LIBS_INIT} ${Poco_LIBRARIES})
```



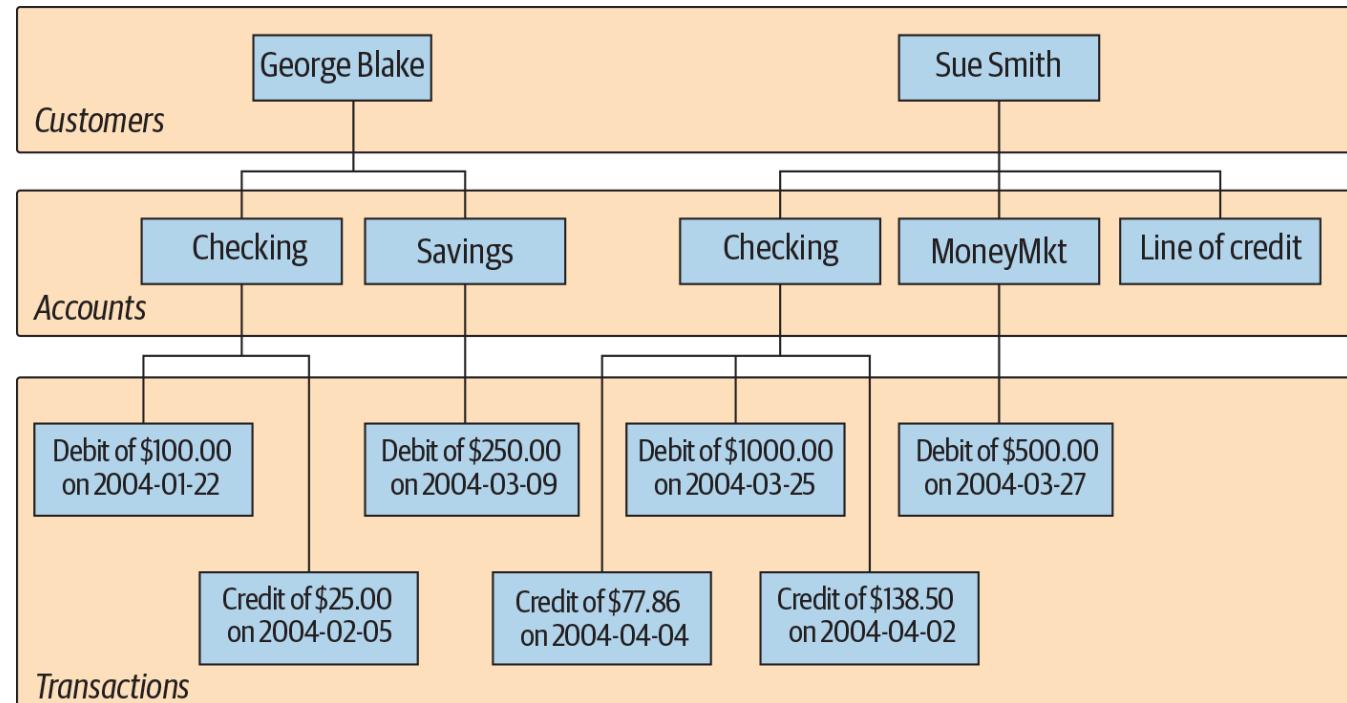
MariaDB

[HTTPS://MARIADB.ORG/](https://mariadb.org/)

Системы управления базами данных

- Хранят данных на долговременных носителях
- Предоставляют интерфейс по созданию данных, поиску и получению данных, обновлению данных и удалению данных
- Различаются по способу хранения и работы с данными
 - noSql
 - SQL

Пример поSQL иерархической базы данных



Пример SQL: реляционных данных

Customer		
cust_id	fname	lname
1	George	Blake
2	Sue	Smith

Account			
account_id	product_cd	cust_id	balance
103	CHK	1	\$75.00
104	SAV	1	\$250.00
105	CHK	2	\$783.64
106	MM	2	\$500.00
107	LOC	2	0

Product	
product_cd	name
CHK	Checking
SAV	Savings
MM	Money market
LOC	Line of credit

Transaction				
txn_id	txn_type_cd	account_id	amount	date
978	DBT	103	\$100.00	2004-01-22
979	CDT	103	\$25.00	2004-02-05
980	DBT	104	\$250.00	2004-03-09
981	DBT	105	\$1000.00	2004-03-25
982	CDT	105	\$138.50	2004-04-02
983	CDT	105	\$77.86	2004-04-04
984	DBT	106	\$500.00	2004-03-27

Основные термины

- **Сущность**
объект реального мира, который интересно хранить в СУБД
- **Колонка**
Атрибут сущности, который сохраняется в таблице
- **Строка**
Набор атрибутов сущности, хранящихся в колонках и представляющие все данные сущности
- **Таблица**
набор строк, которые хранятся на долговременном носителе
- **Основной ключ (primary key)**
Одна или несколько колонок, с помощью которой можно идентифицировать сущность
- **Внешний ключ (foreign key)**
Одна или несколько колонок, с помощью которой можно идентифицировать сущность связанную с текущей, но находящейся в другой таблице

SQL

- Язык для работы с реляционными данными
- Позволяет работать с таблицами (создавать, менять, удалять) и данными

Пример SQL создание таблицы

```
CREATE TABLE corporation (corp_id SMALLINT,  
name VARCHAR(30), CONSTRAINT pk_corporation  
PRIMARY KEY (corp_id) );
```

<https://www.w3schools.com/sql/>

Пример SQL добавление строки в таблицу

```
INSERT INTO corporation (corp_id, name)  
VALUES (27, 'Acme Paper Corporation');
```

<https://www.w3schools.com/sql/>

Пример SQL поиск данных в таблице

```
SELECT name FROM corporation WHERE corp_id =  
27;
```

<https://www.w3schools.com/sql/>

Пример SQL ПОИСК СВЯЗАННЫХ данных в таблицах

```
SELECT t.txn_id,  
t.txn_type_cd,  
t.txn_date, t.amount  
FROM individual i  
INNER JOIN account a ON  
i.cust_id = a.cust_id  
INNER JOIN product p ON  
p.product_cd =  
a.product_cd  
INNER JOIN transaction  
t ON t.account_id =  
a.account_id  
WHERE i.fname =  
'George' AND i.lname =  
'Blake' AND p.name =  
'checking account';
```

Customer		
cust_id	fname	lname
1	George	Blake
2	Sue	Smith

Account			
account_id	product_cd	cust_id	balance
103	CHK	1	\$75.00
104	SAV	1	\$250.00
105	CHK	2	\$783.64
106	MM	2	\$500.00
107	LOC	2	0

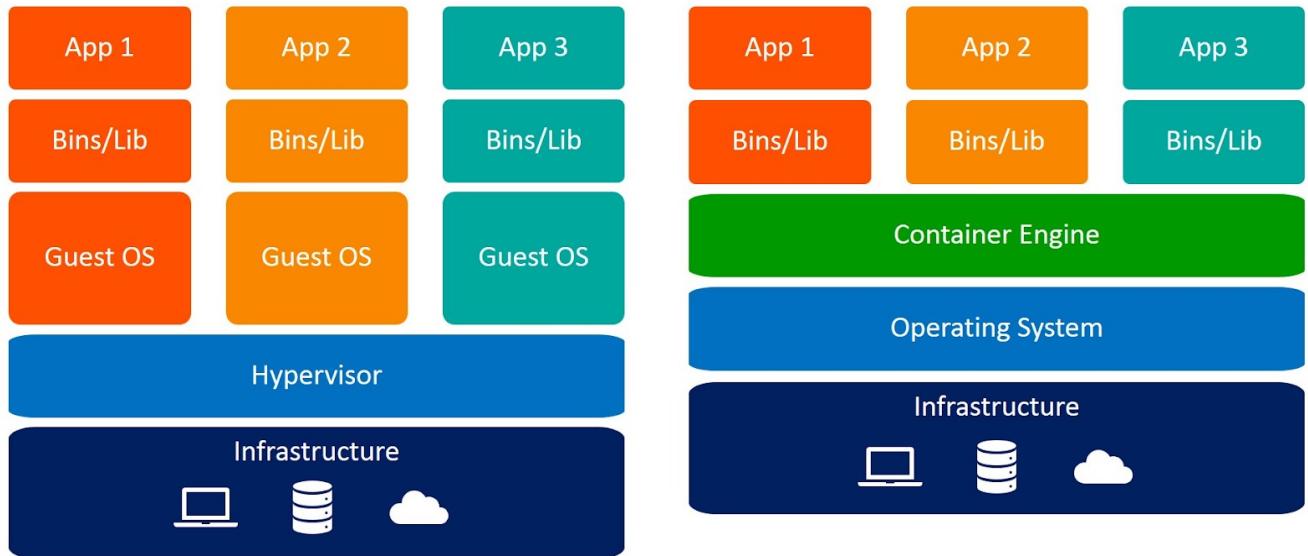
Product	
product_cd	name
CHK	Checking
SAV	Savings
MM	Money market
LOC	Line of credit

Transaction				
txnid	txn.type_cd	account_id	amount	date
978	DBT	103	\$100.00	2004-01-22
979	CDT	103	\$25.00	2004-02-05
980	DBT	104	\$250.00	2004-03-09
981	DBT	105	\$1000.00	2004-03-25
982	CDT	105	\$138.50	2004-04-02
983	CDT	105	\$77.86	2004-04-04
984	DBT	106	\$500.00	2004-03-27



Docker

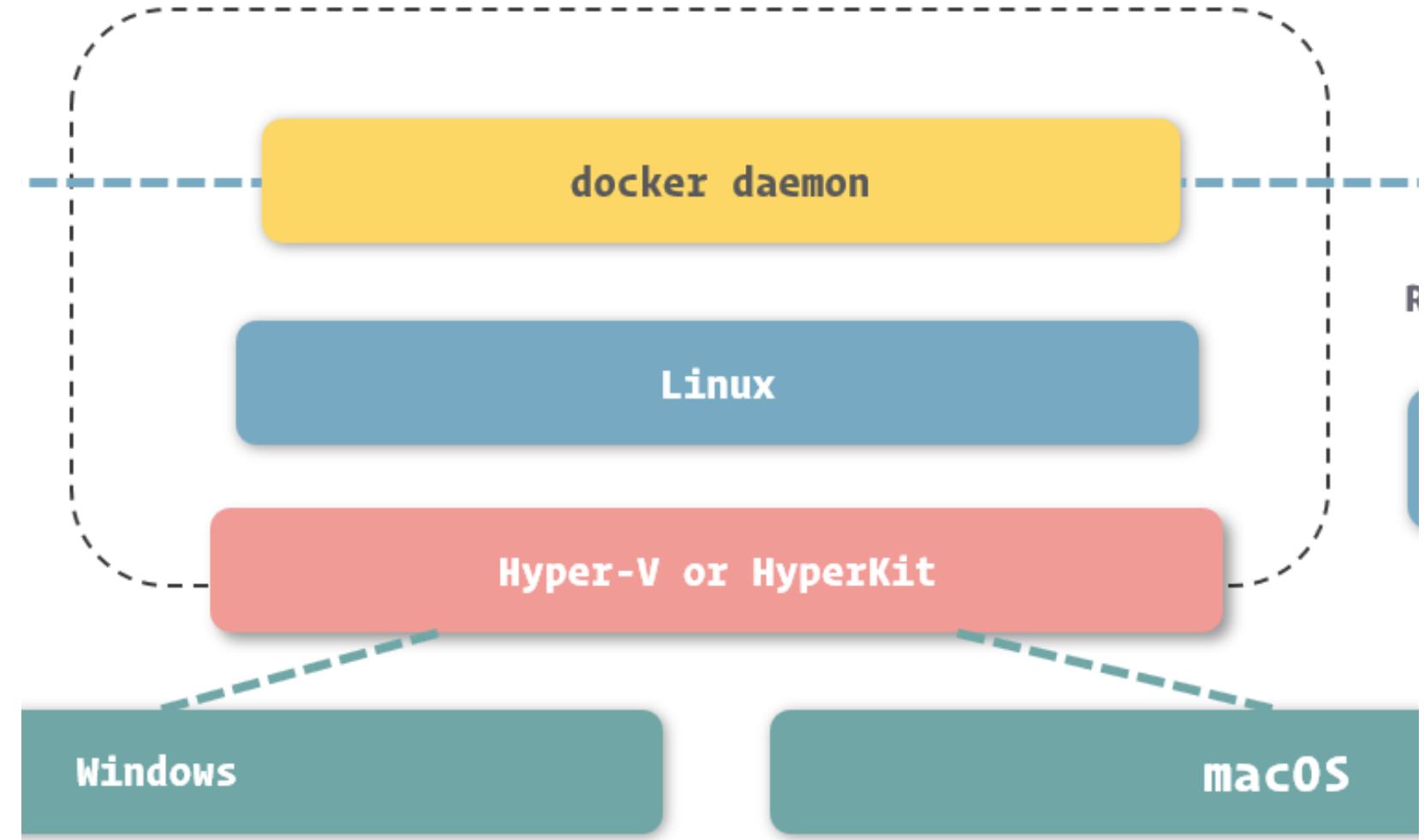
Контейнеры



Virtual Machines

Containers

Есть ли не на linux?



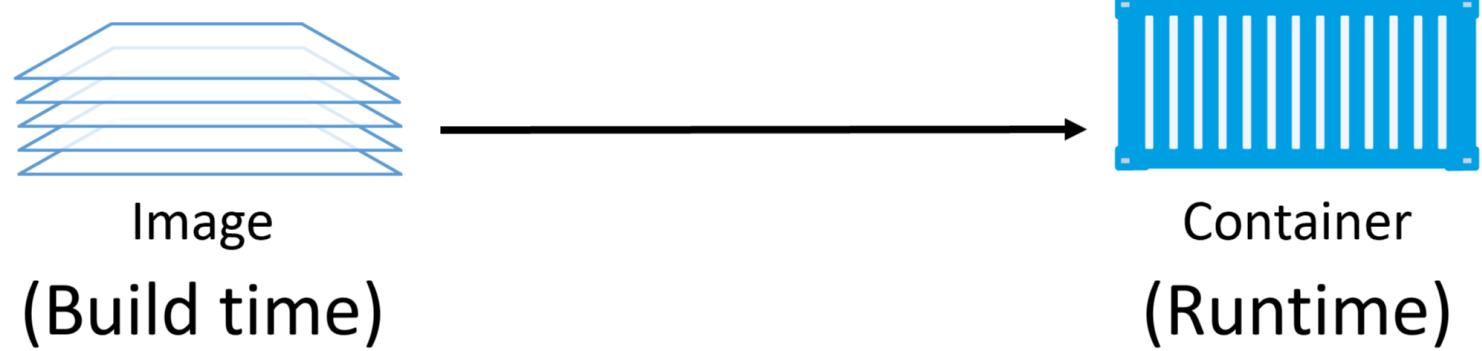
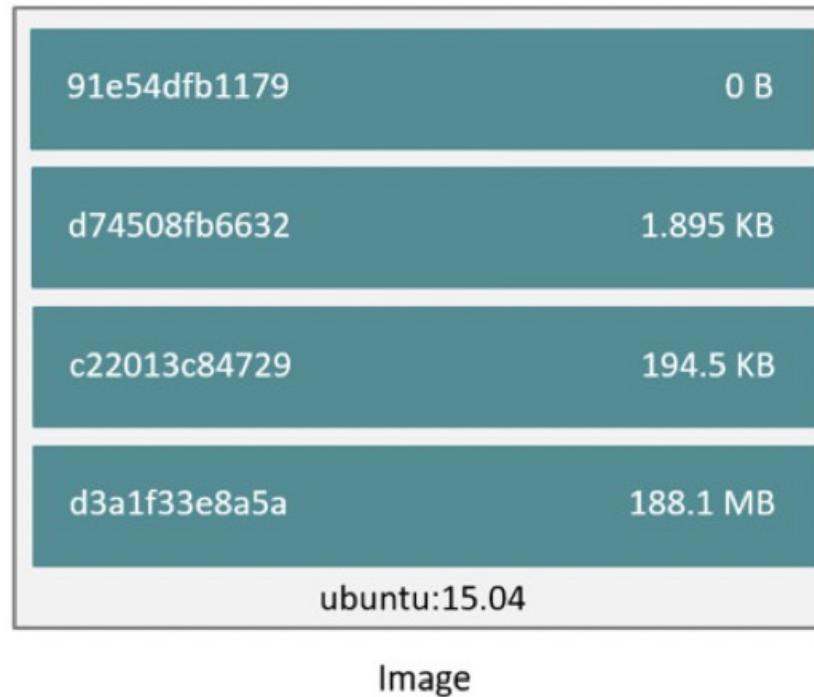


Image & Container

Dockerfile

Текстовый файл с build инструкциями (для сборки образа)
Инструкции декларативно описывают image
Каждая инструкция – промежуточный image
Сборку image делает docker ci



CMD

RUN

RUN

ADD/COPY

пример

```
# Use an official Python runtime as a parent image
FROM python:3.7-slim

# Set the working directory in the container to /app
WORKDIR /app

# Add the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Docker Compose

- Пакетный менеджер (по аналогии с composer и прт, только у docker — контейнеры), позволяющий описывать необходимую структуру в одном файле (конфиге).
- <https://docs.docker.com/compose/compose-file/compose-file-v3/>

пример

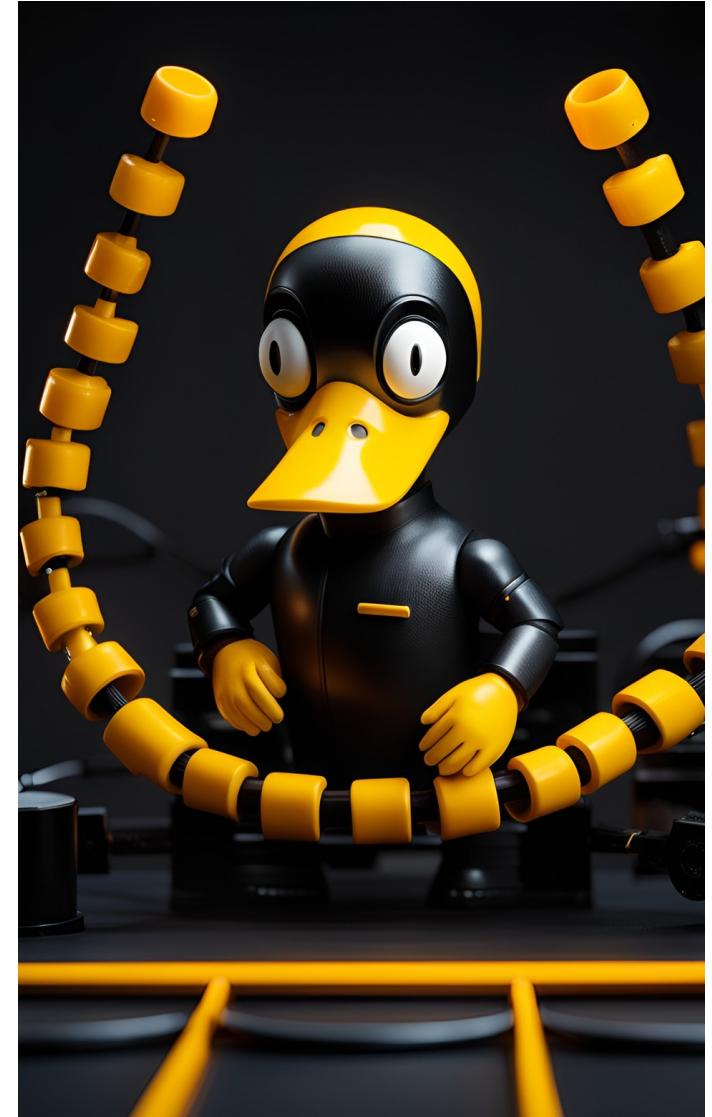
```
version: '3'  
services:  
  web:  
    build:  
      context: .  
      dockerfile: Dockerfile  
    ports:  
      - "8080:80"
```

Как работает?

- Описываем **docker-compose.yml**
- Запускаем **docker-compose up --build**

Пример

о1_Docker
запуск простого приложения в docker

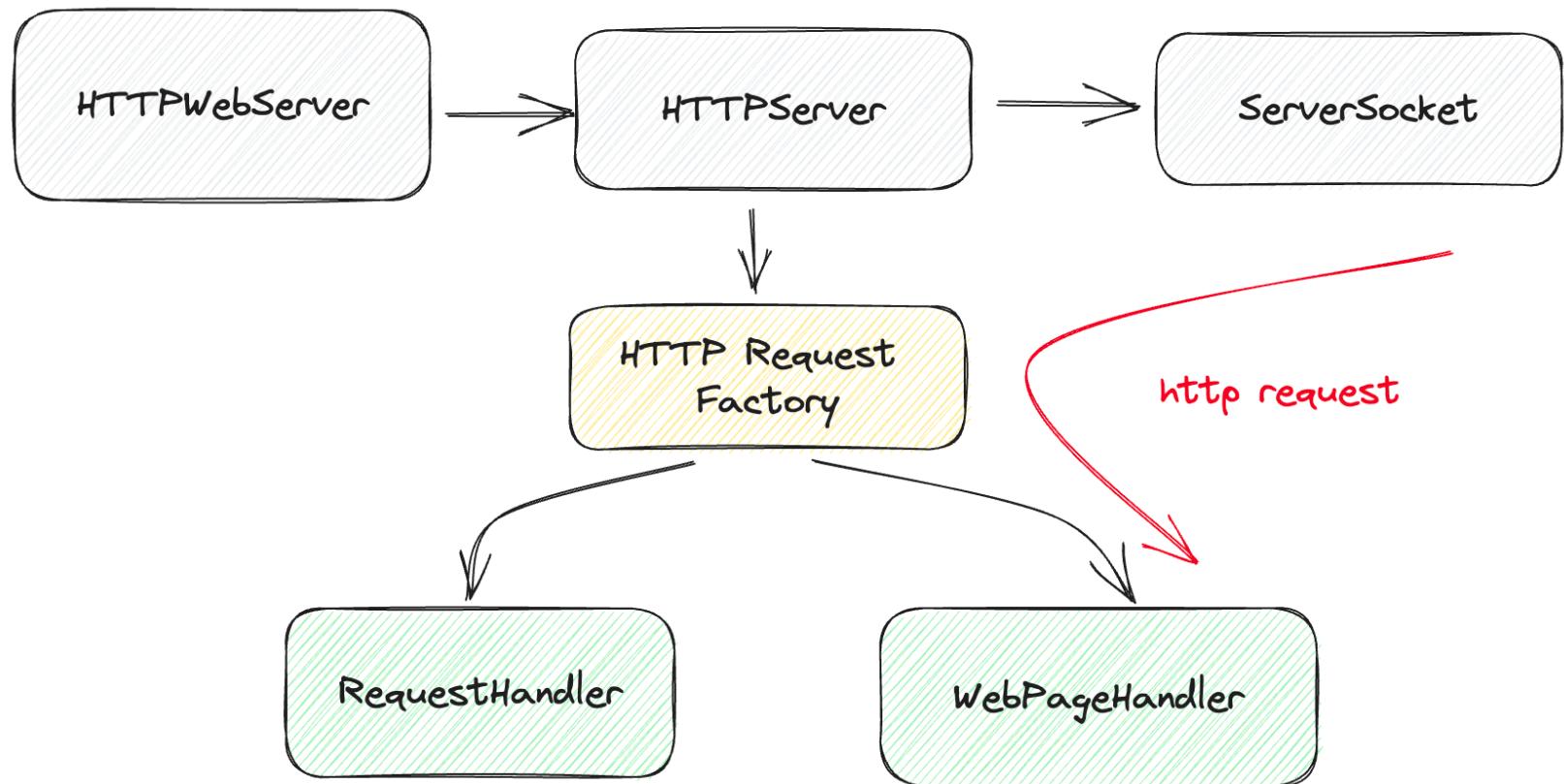


Простое веб-приложение



Логика построения сервера

классы





Пример

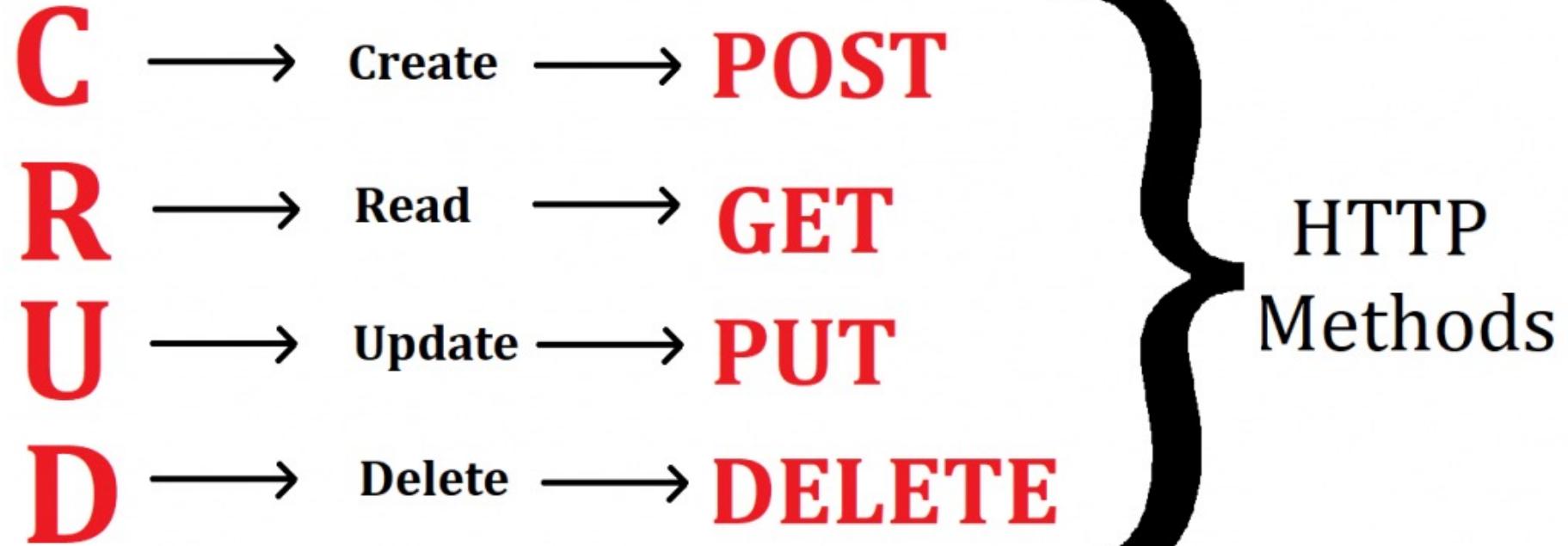
простое веб-приложение



REST

REST

- Является **архитектурным стилем**
- REST работает с **ресурсами по протоколу HTTP**
- Ресурсом может быть любая целостная и осмысленная структура, к которой можно обратиться
- В основе REST лежит **идентификация каждого ресурса с помощью URL**
- **REST** позволяет создать ресурс, удалить ресурс, получить или изменить ресурс



Интерпретация методов HTTP

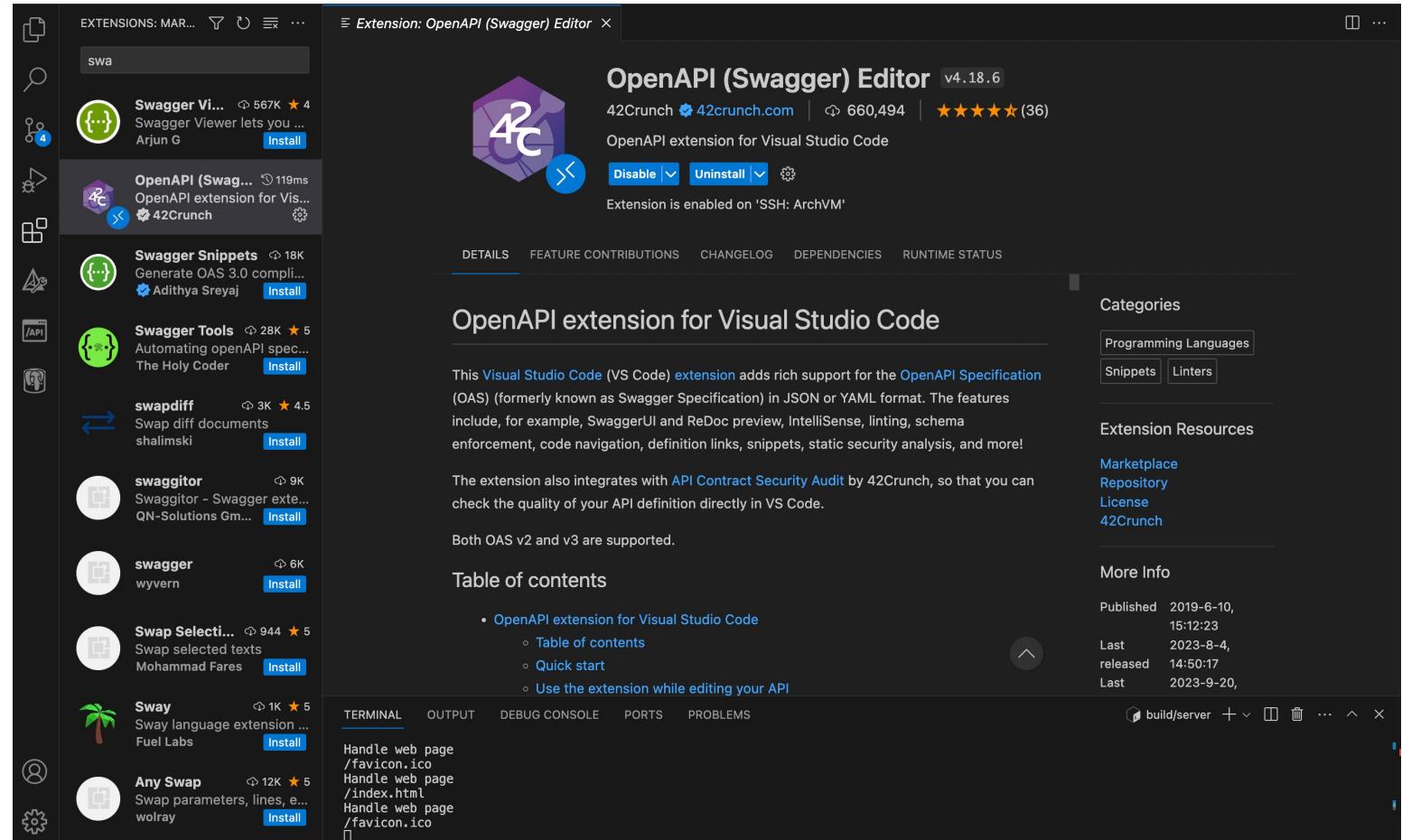
Open API

<https://swagger.io/docs/specification/about/>

Пример

```
openapi: '3.0.0'
info:
  version: '1.0.0'
  title: 'Sample Server'
  description: Demo of REST API
servers:
  - url: http://192.168.64.6:8080
    description: Sample server
paths:
  /request:
    parameters:
      - in: query
        name: session_id
        description: The unique identifier of some session
        required: true
    schema:
      $ref: '#/components/schemas/SessionId'
  get:
    summary: Read a session
    responses:
      '200':
        description: The session corresponding to the provided
`sessionId`
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Sessions'
```

OpenAPI (Swagger) Editor plugin kVSCode



Пример

простое веб-приложение с REST API

Взаимодействие с СУБД в Роско



Пример

ПОЛНОЦЕННЫЙ СЕРВИС



Генерируем данные

The screenshot shows the 'Online data generator' interface with the following sections:

- Step 1: Define columns**

Column Name	Field Type		
first_name	First Name	<input checked="" type="checkbox"/>	<input type="checkbox"/>
last_name	Last Name	<input checked="" type="checkbox"/>	<input type="checkbox"/>
email	Email Address	<input checked="" type="checkbox"/>	<input type="checkbox"/>
title	Job Title	<input checked="" type="checkbox"/>	<input type="checkbox"/>
+ Add column		Reset columns	
- Step 2: Export data**

Rows: Export Type: EXPORT

Choose how many rows do you want to generate

Choose how your data will look like
- Step 3: Preview data live**

A large gray placeholder area where data would be previewed.

A cookie consent banner is visible on the right side:

Our website use cookies to ensure you get the best experience on our website!
Learn more [Got it!](#)

Генерируем данные

<https://www.onlinedatagenerator.com/>

Пример

загружаем данные

Тестируем производительность

[HTTPS://GITHUB.COM/WG/WRK](https://github.com/wg/wrk)





Спасибо!
НА СЕГОДНЯ ВСЕ
