

# **Введение**

**Матвей Попов**

**Кто я такой?**

# **Матвей Попов**

**Team Lead at Avito**

- Более 5 лет коммерческого опыта
- Начинал как Android developer
- Работал в аутсорс, маленьких компаниях и больших Enterprise
- Руководжу кросс-функциональной командой
- Авторский курс в OTUS
- Пишу код с 10 лет

# **План курса**

1. Сначала поговорим глобально про программирование и его историю
2. Обсудим JVM, как устроена работа вокруг Java
3. Погрузимся в Kotlin, коснемся concurrency
4. Frameworks. Spring, Ktor, Quarkus
5. Database Access layer
6. Обсудим инфраструктуру вокруг backend-а
7. Поговорим про архитектуру
8. Протоколы и подходы к разработке приложений
9. Очереди, кэши, мониторинг, алерты, логирование

**Как будет складываться  
оценка?**

- Хотим ли мы экзамен?
- Оценка за домашки - средняя по всем
- Итоговая - ?

**Что нам понадобиться?**

- Github. Мой: <https://github.com/Ferum-bot>
- IntelliJ IDEA последней версии
- Время на выполнение домашек 😬
- Предпочтение Unix base operating system
- Пары записываются
- Вопросы?

**А зачем этот курс нужен?**

- Чтобы понять, что Kotlin это 😎
- Широко посмотреть на программирование
- Узнать из чего глобально состоит разработка почти любого приложения

**Что я хочу от вас?**

- Не стесняться задавать вопросы
- Делать домашки(ну пожалуйста)
- Наслаждаться

# План на урок

- Краткая(очень) история разработки
- Что такое Web и как он устроен?
- Что такое backend? Кто такие backend разработчики?
- Backend это очень круто!

# **Краткая история разработки ПО**

Ну реально краткая....

# Ада Лавлейс

## Первая компьютерная программа

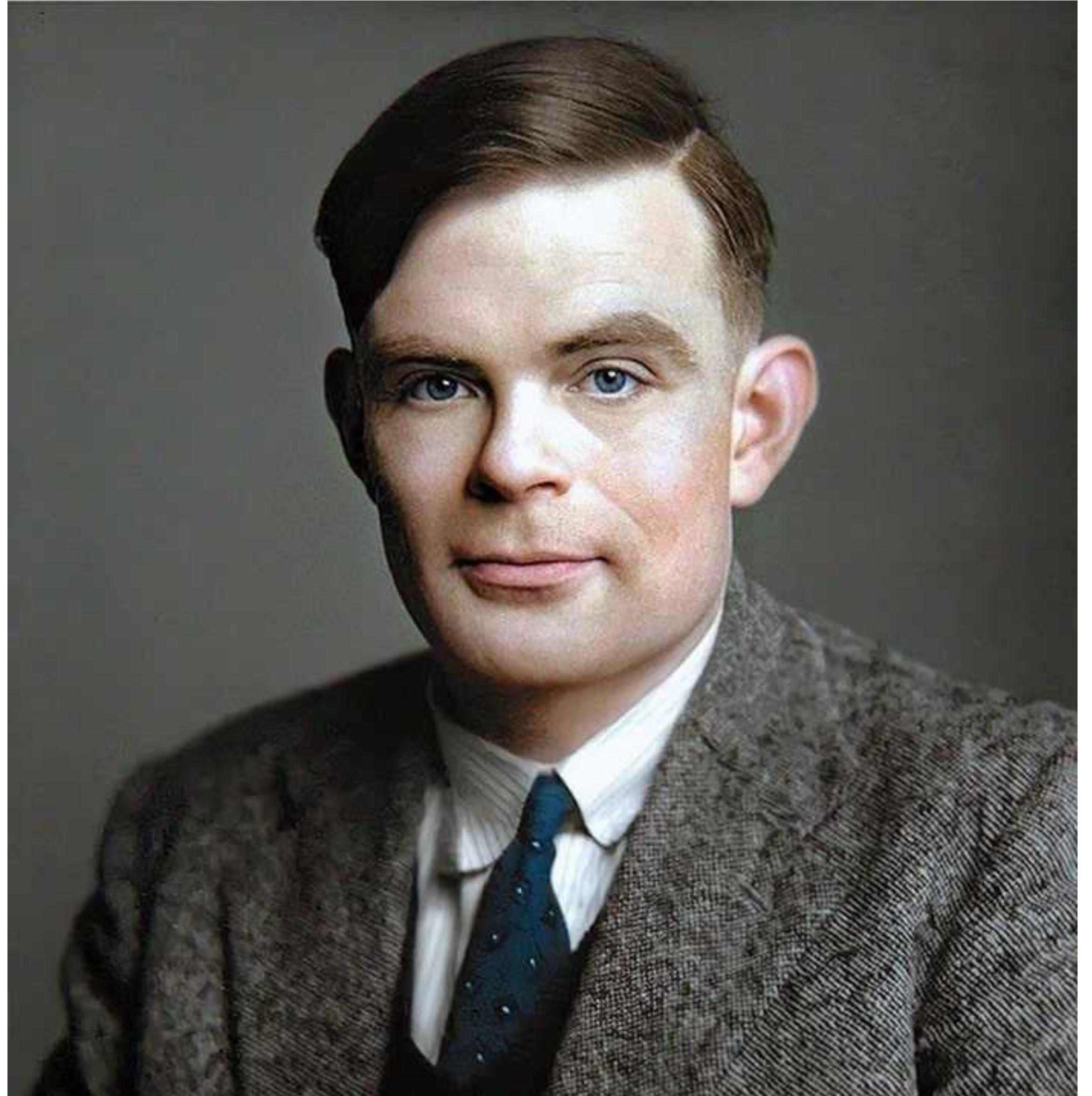
- Чарльз Бэббидж и проект “аналитическая машина” в 1833
- Ада Лавлейс написала алгоритм вычисления последовательности численности Бернулли



# Алан Тьюринг

## Математик и криптограф

- В 1936 впервые описал понятие “Алгоритм”
- Предложил абстрактную вычислительную машину
- Благодаря Тьюрингу зародилась кибернетика



# Кибернетика

Наука об общих закономерностях получения, хранения, преобразования и передачи информации в сложных управляющих системах, будь то машины, живые организмы или общество.<sup>[2]</sup>

**Теперь о программировании**

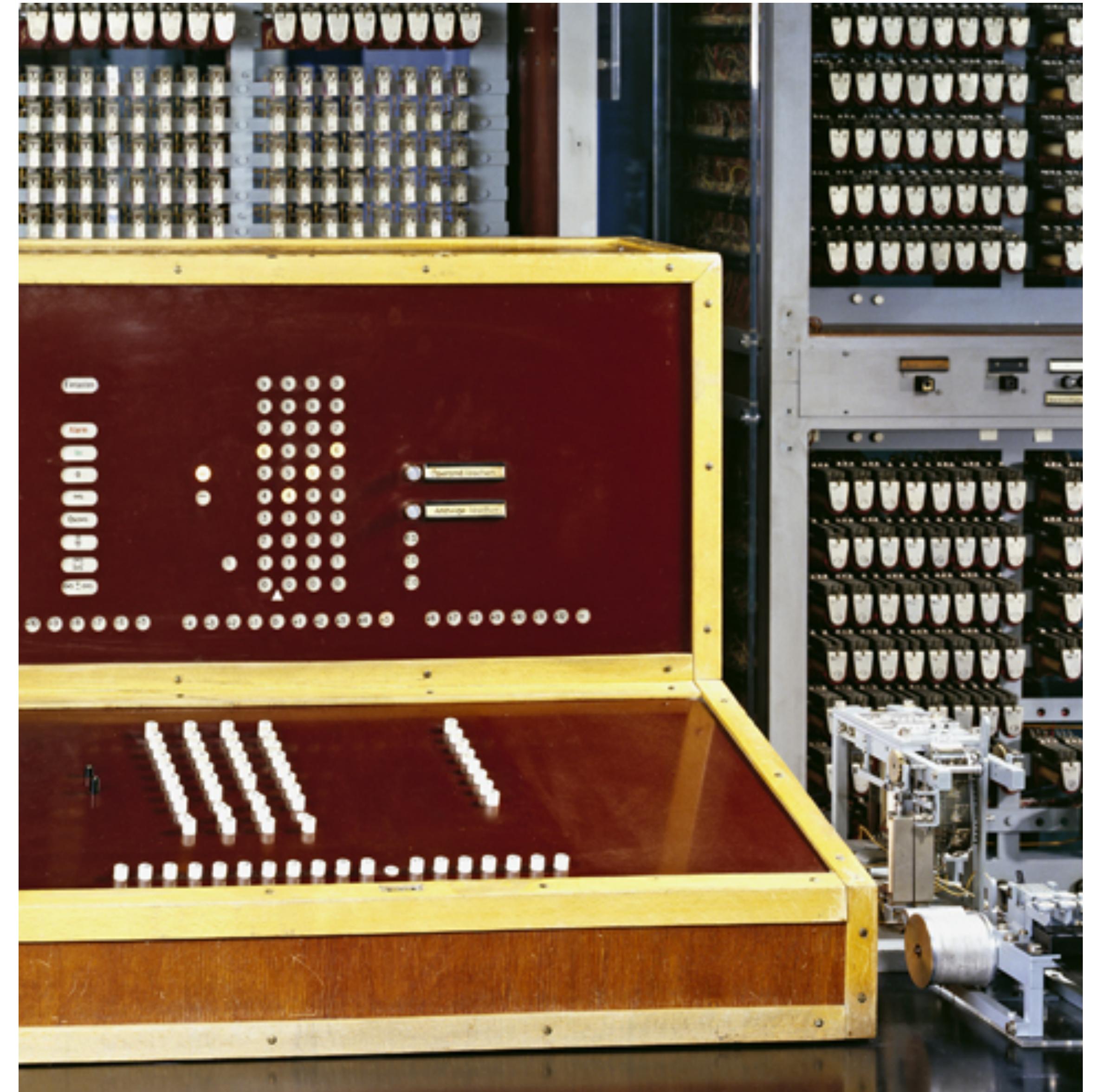
**Без чего невозможно  
программировать?**

**без компьютера!**

# Z3 (1941)

Первый работающий компьютер

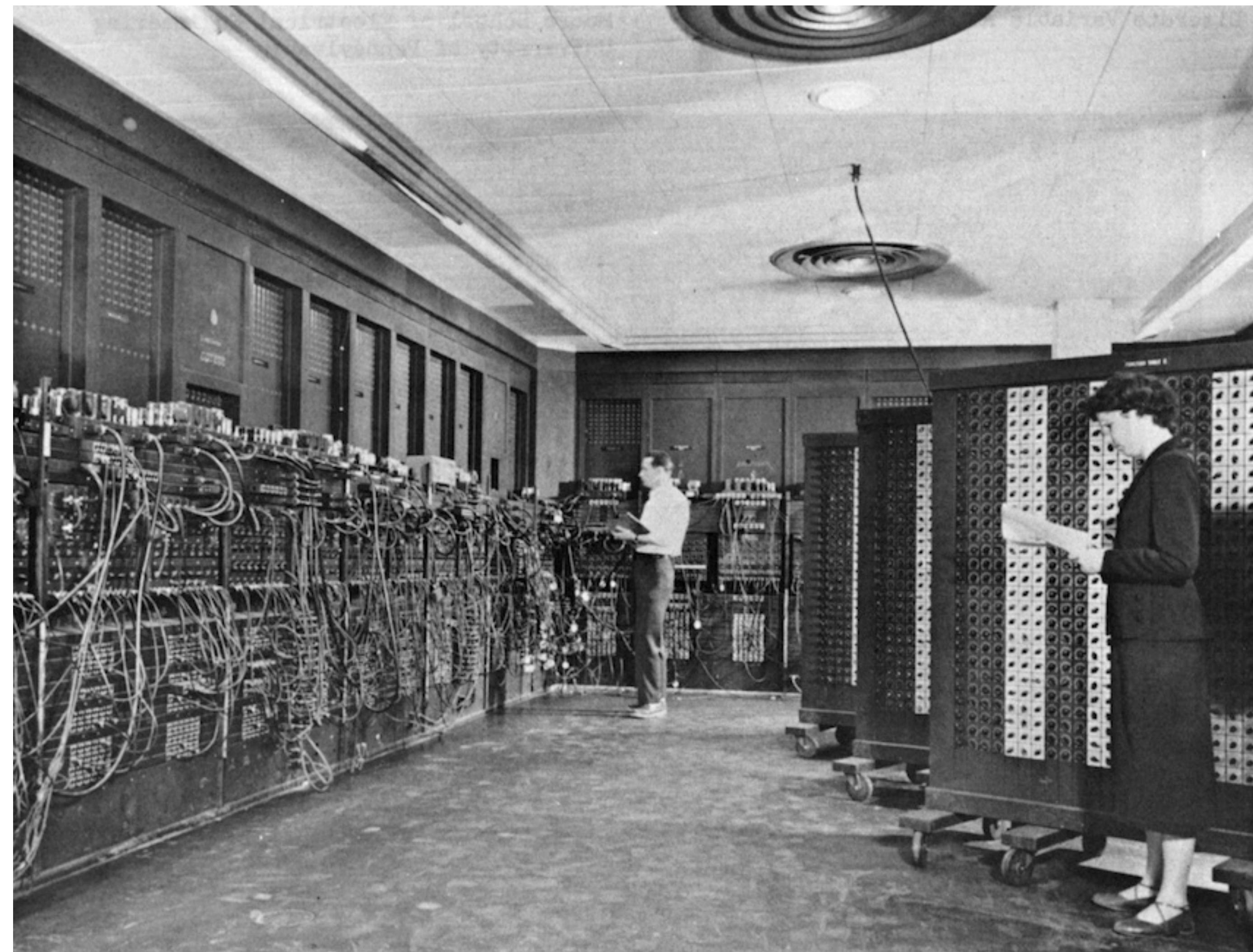
- Полноценный электромагнитный компьютер
- Имел двоичную системы счисления
- Секретный проект немецкого правительства
- Оригинал разрушен в 1943



# ENIAC (1946)

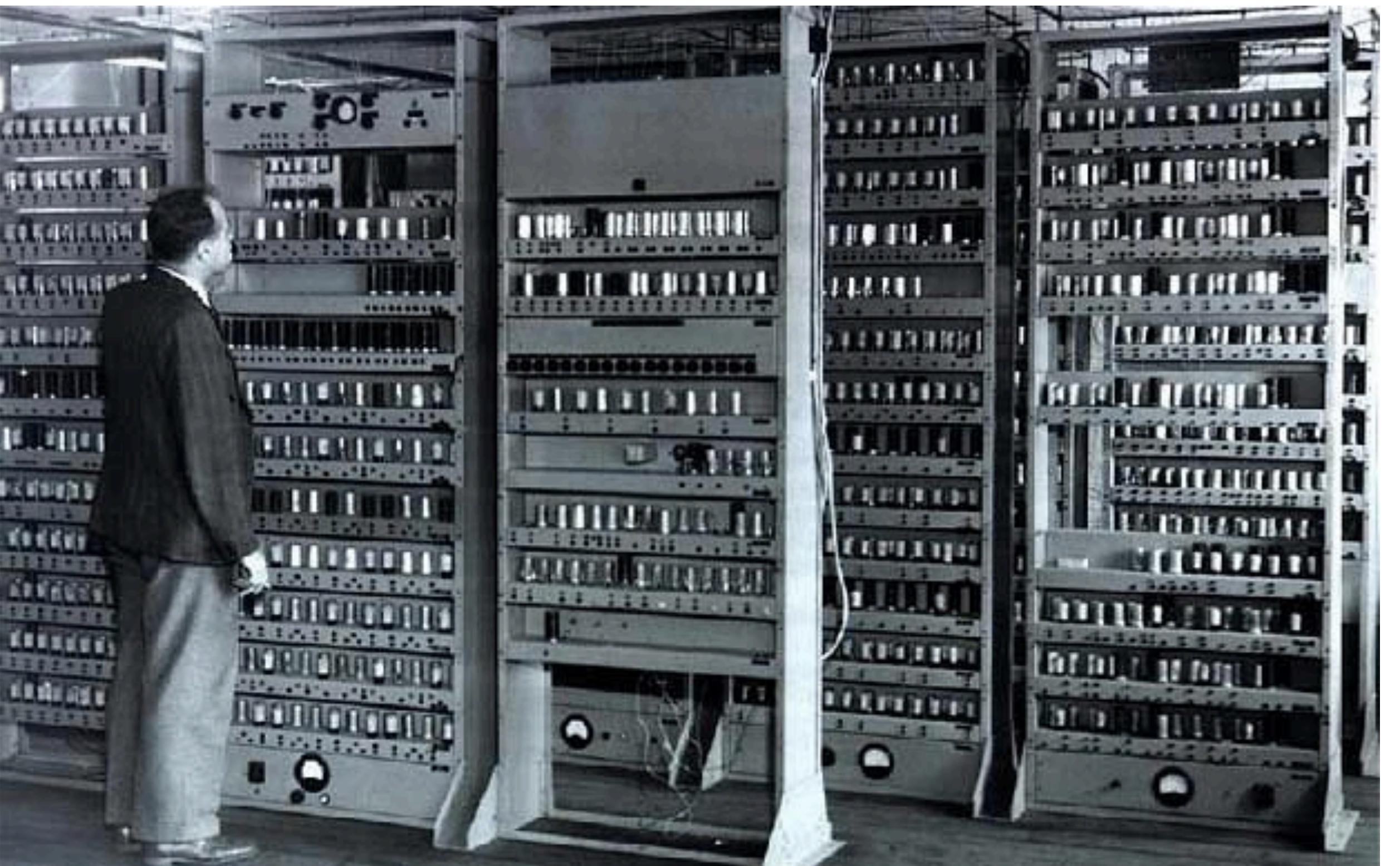
## ЭВМ общего назначения

- Возможность перепрограммировать
- Создан для американской армии
- 150 футов в ширину
- На программирование задачи уходило несколько дней



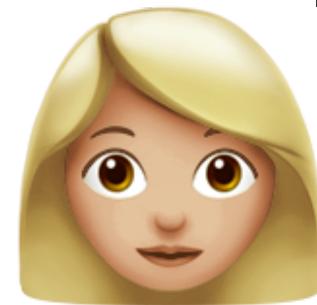
Это конечно круто, но что там  
с языками? 

- До 1950-х программы для компьютеров писали люди, которые их непосредственно разрабатывали
- Программа это набор инструкций по доступам к ячейкам памяти
- Код выглядел как последовательность числе



Кто же они, первые  
программисты?

# Женщины



- В NASA на должность “людей-компьютеров” нанимали женщин
- Они рассчитывали самые разные вещи необходимые для запуска космического корабля
- Точные расчеты отправили “Вояджер” исследовать солнечную систему и человека на луну



# Маргарет Хэмилтон



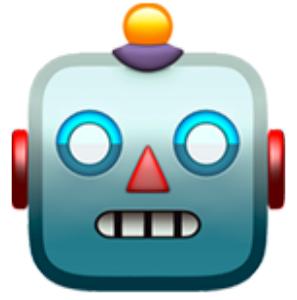
**Кто она?**

- Руководитель команды разработки ПО для “Аполлон”
- Ввела термин “разработка программного обеспечения”
- Внесла невероятный вклад в освоении космоса



**Ну теперь точно о  
программировании**

# Ассемблер



- Первые ассемблеры были созданы к концу 1950-х
- Язык ассемблера – представление команд процессора в виде, доступным для чтения человеком
- Привязан к конкретной системе
- По сути – множество команд

```

C000          ORG      ROM+$0000 BEGIN MONITOR
C000 8E 00 70 START    LDS      #STACK

*****
* FUNCTION: INITA - Initialize ACIA
* INPUT: none
* OUTPUT: none
* CALLS: none
* DESTROYS: acc A

0013        RESETA   EQU      %00010011
0011        CTLREG   EQU      %00010001

C003 86 13  INITA    LDA A  #RESETA  RESET ACIA
C005 B7 80 04           STA A  ACIA
C008 86 11           LDA A  #CTLREG  SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A  ACIA

C00D 7E C0 F1        JMP     SIGNON  GO TO START OF MONITOR

*****
* FUNCTION: INCH - Input character
* INPUT: none
* OUTPUT: char in acc A
* DESTROYS: acc A
* CALLS: none
* DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH     LDA A  ACIA      GET STATUS
C013 47           ASR A
C014 24 FA           BCC    INCH      RECEIVE NOT READY
C016 B6 80 05           LDA A  ACIA+1  GET CHAR
C019 84 7F           AND A  #$7F  MASK PARITY
C01B 7E C0 79           JMP    OUTCH   ECHO & RTS

*****
* FUNCTION: INHEX - INPUT HEX DIGIT
* INPUT: none
* OUTPUT: Digit in acc A
* CALLS: INCH
* DESTROYS: acc A
* Returns to monitor if not HEX input

C01E 8D F0  INHEX    BSR     INCH      GET A CHAR
C020 81 30           CMP A  #'0  ZERO
C022 2B 11           BMI    HEXERR  NOT HEX
C024 81 39           CMP A  #'9  NINE
C026 2F 0A           BLE    HEXRTS GOOD HEX
C028 81 41           CMP A  #'A
C02A 2B 09           BMI    HEXERR  NOT HEX
C02C 81 46           CMP A  #'F
C02E 2E 05           BGT    HEXERR
C030 80 07           SUB A  #7   FIX A-F
C032 84 0F           HEXRTS AND A  #$0F  CONVERT ASCII TO DIGIT
C034 39             RTS

C035 7E C0 AF  HEXERR  JMP     CTRL      RETURN TO CONTROL LOOP

```

**А что стало после?**

- Появление языков не привязанных к определенной ЭВМ
- Разработка компиляторов под язык

# Компиляция

Трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда язык ассемблера).

# **FORTRAN**

- Разработан в 1957
- Первый язык высокого уровня
- **FOR**mul**A** **T**RAN**S**lator
- Считается первым широко используемым языком программирования

**Заходит в бар ALGOL, LISP и  
COBOL**

**А бармен им говорит:**

**Песок потом за собой уберите**

- Каждый из языков создавался как “улучшение” FORTRAN
- Появлялись новые возможности, ветвления и циклы
- Некоторые языки были направлены на решение узких задач

**А как появились современные  
языки?**

**С - первый популярный язык и  
наследник ALOGOL**

**В 1980-х в программирование  
приходят “простые” люди**

**К чему это приводит?**

# **Появление скриптовых языков**

- Perl - 1987
- Python - 1991
- Ruby - 1993
- Потребность в языках, которые легко может использовать простой человек

**А потом случилось немыслимое**



**Появление и быстрое  
распространение Интернета**

# Интернет

Коммуникационная сеть и всемирная система  
объединённых компьютерных сетей для хранения и  
передачи информации

# **Появление специализированных языков**

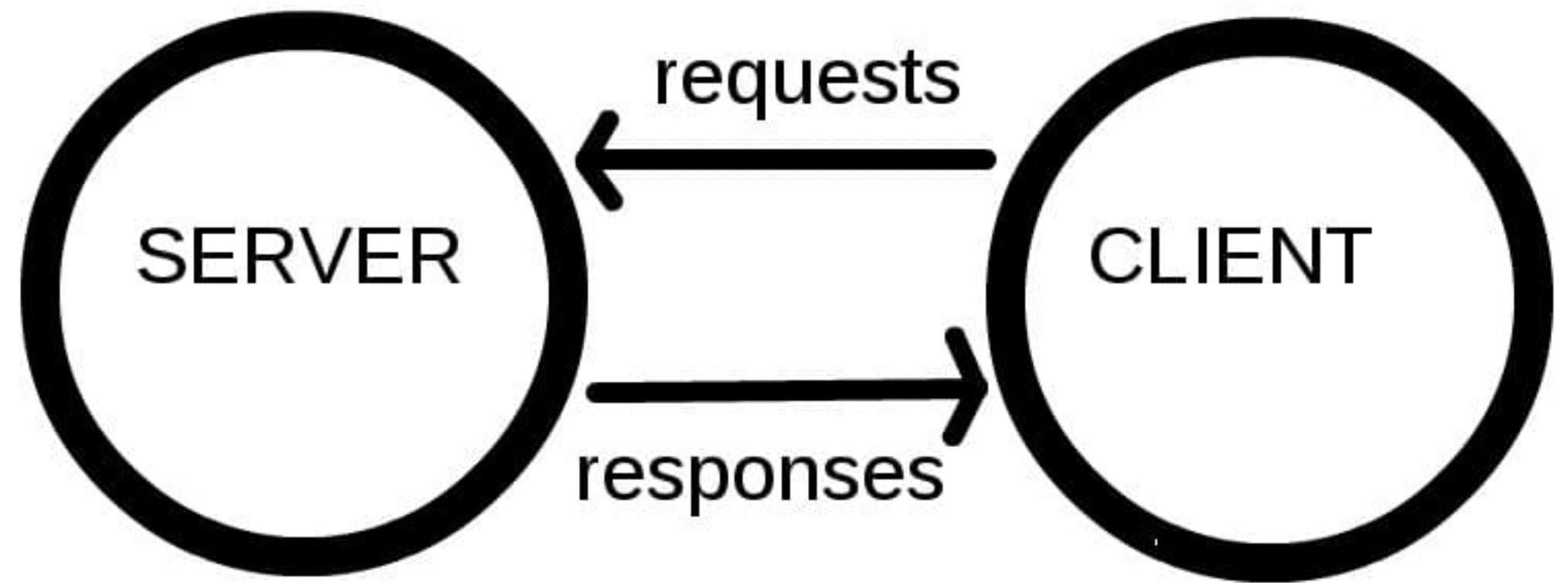
- PHP
- JavaScript
- Java(?)
- Старые языки становились Web ориентированными
- Началось бурное развитие ИТ технологий

**Как устроен современный  
Web?**

**Web == Интернет**

Что под Web обычно  
понимают разработчики?

- Клиент-серверная модель
- Протокол HTTP(S), порты, TCP/IP, DNS
- Браузер, HTML, CSS, JavaScript
- Frontend -> UI, Backend -> Data



# А что там еще есть?

- Множество других сетевых протоколов
- Протоколы обмена данными
- Множество балансировщиков
- VPN, Proxy, шифрование
- Разные сервера, базы данных
- Множество теорем и подходов
- ML, DL
- А еще же есть Web 3.0....

И что же из этого **backend**?

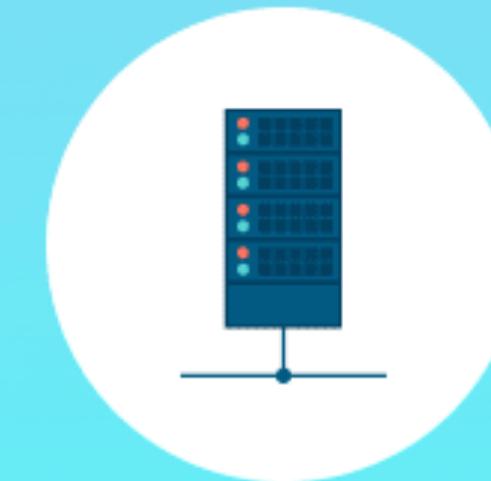
**Правильно - всё!**

# Что обычно понимают под backend?

- Внутренняя и вычислительная логика сервиса
- Всё что относится к программно-аппаратной части сервиса
- Данные: хранение, манипуляция и преобразование
- Обслуживание серверов
- Тестирование, расчет нагрузки и поддержка
- Даже административный кабинет



**System  
architecture**



**Servers**



**Database**



**Data analysis**



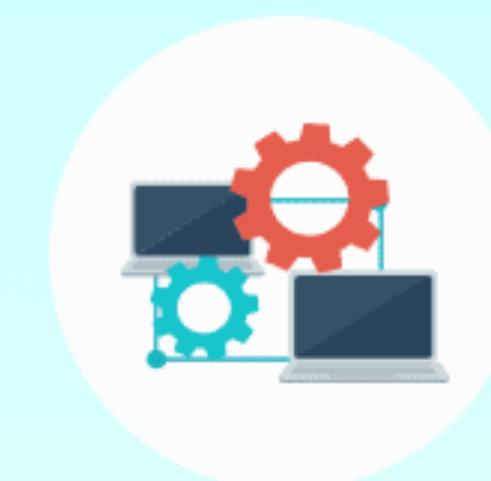
**Security**



**Frameworks**



**Scalability**

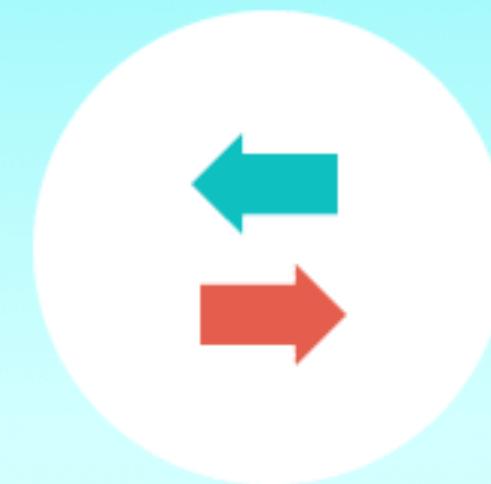


**Operating  
system**



**Business logic**

# BACKEND IS:



**APIs**

Ну а кто такие backend-dev?  
(Who)

- Программисты
- Администраторы баз данных
- Архитекторы
- Сетевые инженеры
- Тестировщики
- Инфраструктурщики

**т.е. Боги?**

**ну почти**

# **Инженеры**

# **Почему backend это круто**

- Множество сфер развития
- Вечная актуальность
- Навыки никогда не устареют
- Понятный вертикальный рост
- Возможность создавать революционные сервисы
- Возможность изобретать собственные продукты

**Куда можно развиваться?**

- Machine Learning если нравится математика и видеокарты
- Operating System dev если нравится думать про память и приоритизация процессов
- DevOps если нравятся деньги
- SmartContracts/Blockchain dev если нравится быть модным
- Networks dev - если нравится быть не таким как все
- Cybersecurity specialist - если хотите быть хакером

# **Полезные материалы**

- Крутая статья про Маргарет Гамильтон: <https://clck.ru/35J3AX>
- Timeline развития Web: <https://clck.ru/35J3CH>
- Тут можно подробнее почитать про LISP, ALGOL, KABOL: <https://clck.ru/DJ8YC>

**Вопросы?**

**Спасибо за уделенное время!**