David Shekunts

🦾 IT-Качалка Давида Шекунца представляет 💪

ООП. От любви до ненависти — один шаг. Но куда?

👋 Привет

David Shekunts

Меня зовут **Давид Шекунц**

Tg – @it-kachalka
Site – about-me.davidshekunts.ru

👋 Привет

David Shekunts

10 лет
в IT

🏦 **Рокетбанк**
(React Team lead)

📡 **Мегафон IIOT**
(CTO)

💳 **Nearpay**
(Senior Node.ts)

🏰 **GoMining**
(Go / Node.ts Tech Lead)

🪨 **GMT Token**
(Senior Node.ts)

🧠 **Deeplay**
(Архитектор)

🎰 **Smartvend**
(Node.ts Tech Lead)

🔞 **Асткол**
(Архитектор)

✨ **Sanctury**
(Go Tech Lead)

⛵ **Three Zeta Studio**
(Основатель)

🎮 **Meveric**
(Основатель / Go Lead)

🦾 **IT-Качалка** 💪
(Тамада)

👋 Стэк

David Shekunts

JS → JQuery → Backbone → CoffeeScript → Angular 1 → TypeScript → Angular 2 → React → PHP Yii → Laravel → Python Django → RoR → C# ASP .NET → Node.js → Go

👋 Стэк

David Shekunts

JS / TS / PHP / Python / Ruby / C# / Node.js / Go

=

Мультипарадигмальность

👋 Мультипарадигмальность

David Shekunts

ООП	ФП	ПП
class	Lambdas	for
extends	HOF	while
private / public	ADT	if
		mutability

👋 Мультипарадигмальность

David Shekunts

1. Набор правил
2. Гибкость дизайна кода

👋 Мультипарадигмальность

David Shekunts

👑 ООП 👑

😢 ФП 😢

👴🏿 ПП 👴🏿

👋 Сомнения

David Shekunts

Что-то с ООП было не так... но что?

👋 Путь

David Shekunts

1. Осознания
2. Попытка в альтернативы
3. Нахождение

👋 Путь

David Shekunts

1. Осознания
2. Попытка в альтернативы
3. Нахождение

David Shekunts

🔍 Осознание 🔍

🔍 Осознание

David Shekunts

Вопрос к аудитории:

```ts
// # 1
class User {
    constructor(
            public email: string;
            public password: string | null;
            public activated: boolean;
    ) {}
}

class UserService {
    static setPassword(user: User, password: string)
{       const hashedPassword = hash(password)

        user.password = hashedPassword;
        user.activated = true;

        return;
    }
}
```

```ts
// # 2
const User = (
        email: string,
        password: string | null,
        activated: boolean,
) => {
    const state = {
        email,
        password,
        activated,
    }

    return {
        setPassword(password: string) {
            const hashedPassword =
hash(password)
            state.password = hashedPassword;
            state.activated = true;

            return;
        }
    }
}
```

🔍 Осознание

David Shekunts

ЧТО такое ООП?

🔍 Осознание

David Shekunts

I. Определение Алана Кея
II. 4 столпа
III. Паттерны и принципы
IV. Синтаксис

🔍 Осознание

David Shekunts

I. **Определение Алана Кея:**

“Программа состоит из ‘объектов’, инкапсулирующих данные и поведение и общающихся сообщениями.”

```ts
class Email {
  private value: string;
  private activatedAt: Date | null;

  activate(activatedAt: Date) {
    this.activatedAt = activatedAt;
  }
}

class User {
  private email: Email;
  private password: string;
  private updatedAt: Date;

  activateByEmail(activatedAt: Date) {
    this.email.activate(activatedAt);
    this.updatedAt = new Date();
  }
}
```

```ts
class Email {
  public value: string;
  public activatedAt: Date | null;
}

class User {
  public email: Email;
  public password: string;
  public updatedAt: Date;
}

function activateByEmail(
    user: User,
    activatedAt: Date,
) {
  user.email.activatedAt = activatedAt;
  user.updatedAt = new Date();
}
```

🔍 Осознание

David Shekunts

II. 4 Столпа

1. Объектная Инкапсуляция
2. Наследование
3. Объектный Полиморфизм
4. Абстракция

🔍 Осознание

David Shekunts

1. Инкапсуляция == Сокрытие

Объектная

Публичные и
приватные методы
“объектов”

Натуральная

- Стороннее API
- Процессы ядра
- Замыкание

🔍 Осознание

David Shekunts

2. Наследование

🔍 Осознание

David Shekunts

3. Полиморфизм

Объектный

Вытекает из Наследования

Интерфейсный

Для полиморфизма используется структурная типизация

```ts
class User {
  private email: Email;
  private password: string;
  private updatedAt: Date;

  activateByEmail(activatedAt: Date) {
    this.email.activate(activatedAt);
    this.updatedAt = new Date();
  }
}

class Admin extends User {
  // ...
}

class Engineer extends User {
  // ...
}

function someFn(user: User) {
  // ...
}

```

```ts
class Admin {
  // ...

  activateByEmail(activatedAt: Date) {
    // ...
  }
}

class Engineer {
  // ...

  activateByEmail(activatedAt: Date) {
    // ...
  }
}

interface Activator {
  activateByEmail(activatedAt: Date): void;
}

function someFn(activator: Activator) {
  // ...
}
```

🔍 Осознание

David Shekunts

4. Абстракция

🔍 Осознание

David Shekunts

III. Паттерны & Принципы

- Fabric
- Abstract Fabric
- Builder
- Prototype

SOLID, GRASP, etc.

🔍 Осознание

David Shekunts

IV. Синтаксис

class / private / public / abstract / etc.

🔍 Осознание

David Shekunts

I. Определение Алана Кея
II. 4 столпа
III. Паттерны и принципы
IV. Синтаксис

🔍 Осознание

David Shekunts

А без чего ООП (не)возможно?

I. Определение Алана Кея
II. 4 столпа
III. Паттерны и принципы
IV. Синтаксис

🔍 Осознание

David Shekunts

1. Применимо ли это определение Алана Кея не только к кодовой базе?

2. Обязан ли я следовать всем 4-м столпам?

3. Обязаны ли мы использовать паттерны и принципы ООП?

4. Могу ли я написать программу, которая следует принципу Алана Кея, столпам и паттернам не используя синтаксиса?

🔍 Осознание

David Shekunts

i. Применимо ли это определение Алана Кея не только к кодовой базе?

“**X** состоит из ‘объектов’, инкапсулирующих данные и поведение и общающихся сообщениями.”

🔍 Осознание

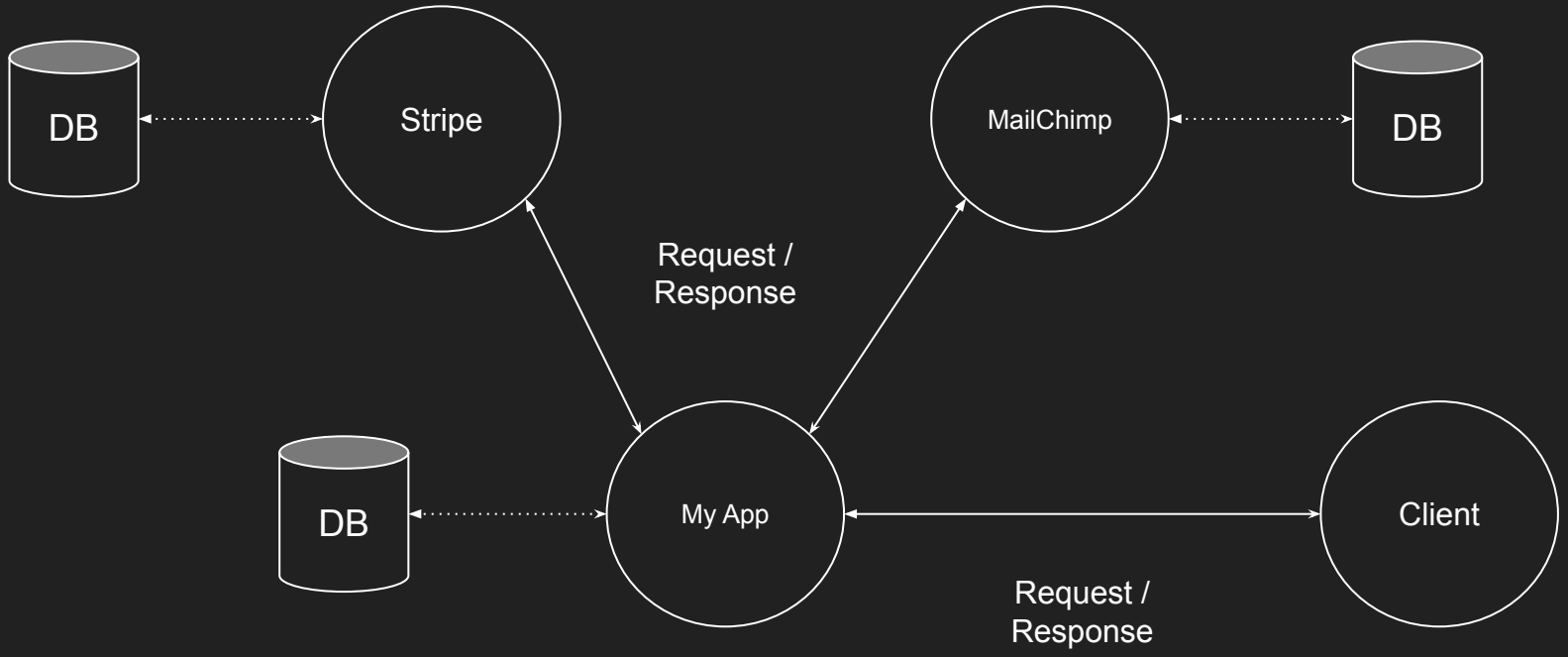
David Shekunts

Вопрос к аудитории

🔍 Осознание

David Shekunts

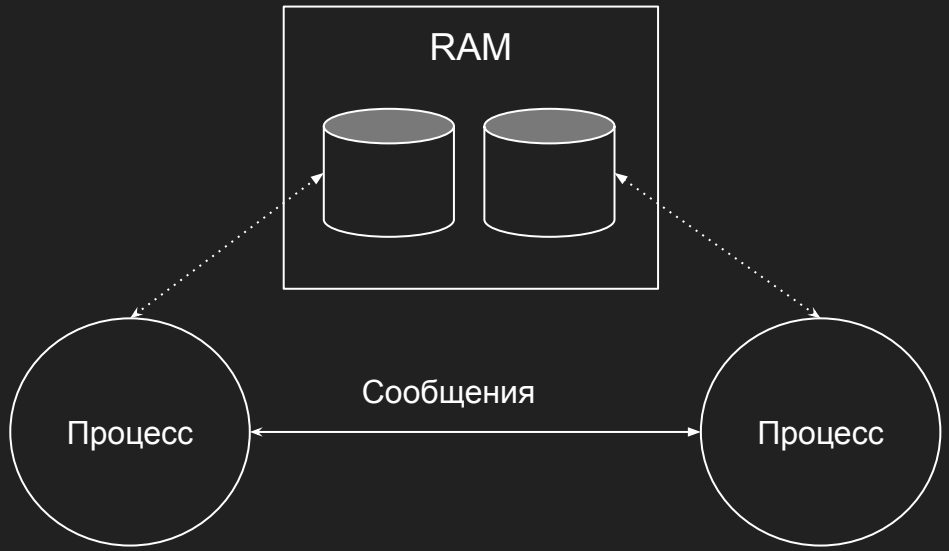
API



🔍 Осознание

David Shekunts

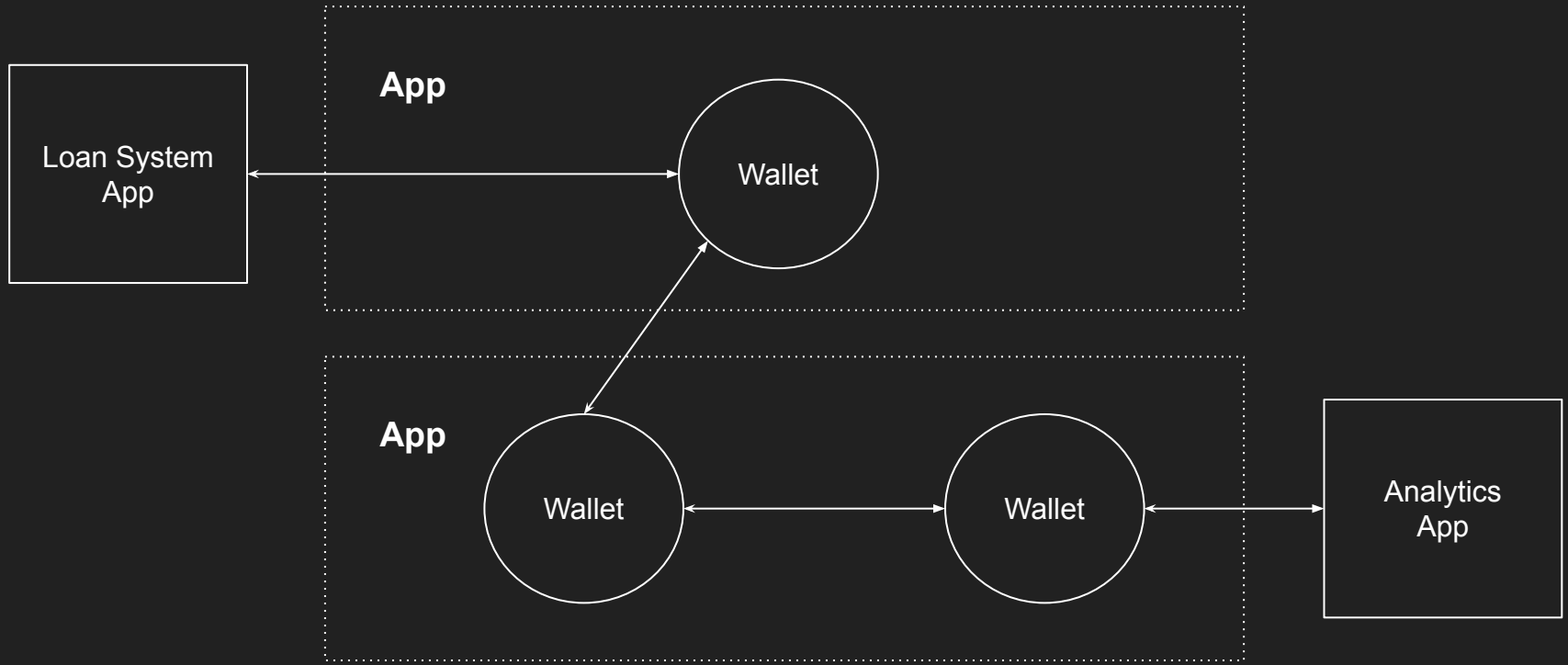
Процесс



👋 !!! УПРОСТИТЬ

David Shekunts

Actor Model



🔍 Осознание

David Shekunts

i. Применимо ли это определение Алана Кея не только к кодовой базе?

Да:

1. Натуральная характеристика множества систем
2. Может лежать в основе кастомных архитектур

🔍 Осознание

David Shekunts

II. Обязаны ли мы следовать всем 4-м столпам?

🔍 Осознание

David Shekunts

Вопрос к аудитории

🔍 Осознание

David Shekunts

II. Обязаны ли мы следовать всем 4-м столпам?

🔴 Абстракция

🔴 Объектный полиморфизм → Интерфейсный

🔴 Наследование → Композиция

🟢 Объектная инкапсуляция

```ts
class Email {
  private value: string;
  private activatedAt: Date | null;

  activate(activatedAt: Date) {
    this.activatedAt = activatedAt;
  }
}

class User {
  private email: Email;
  private password: string;
  private updatedAt: Date;

  activateByEmail(activatedAt: Date) {
    this.email.activate(activatedAt);
    this.updatedAt = new Date();
  }
}
```

```ts
class Email {
  public value: string;
  public activatedAt: Date | null;
}

class User {
  public email: Email;
  public password: string;
  public updatedAt: Date;
}

function activateByEmail(
    user: User,
    activatedAt: Date,
) {
  user.email.activatedAt = activatedAt;
  user.updatedAt = new Date();
}
```

🔍 Осознание

David Shekunts

II. Обязаны ли мы следовать всем 4-м столпам?

Обязаны следовать Объектной инкапсуляции

🔍 Осознание

David Shekunts

III. Обязаны ли мы использовать паттерны и принципы?

🔍 Осознание

David Shekunts

Вопрос к аудитории

🔍 Осознание

David Shekunts

III. Обязаны ли мы использовать паттерны и принципы?

Нет, но рано или поздно мы к ним придем.

🔍 Осознание

David Shekunts

IV. Могу ли я написать программу, которая следует принципу Алана Кея, столпам и паттернам не используя синтаксиса?

🔍 Осознание

David Shekunts

Вопрос к аудитории

```
// # 2
const User = (
        email: string,
        password: string | null,
        activated: boolean,
) => {
    const state = {
        email,
        password,
        activated,
    }

    return {
        setPassword(password: string) {
            const hashedPassword =
hash(password)
            state.password = hashedPassword;
            state.activated = true;

            return;
        }
    }
}
```

👋 Осознание

David Shekunts

1. Применимо ли это определение Алана Кея не только к кодовой базе?
   Да → Выходит за рамки конкретного ЯП

2. Обязан ли я следовать всем 4-м столпам?
   Объектная Инкапсуляция обязательна → Корень ООП

3. Обязаны ли мы использовать паттерны и принципы ООП?
   Нет → Они надстройка над ООП

4. Могу ли я написать программу, которая следует принципу Алана Кея, столпам и паттернам не используя синтаксиса?
   Да → ООП Синтаксис != “ООП”

👋 Осознание

David Shekunts

Существует 3 вида ООП

👋 Осознание

David Shekunts

3 вида ООП

1. **ООП как архитектура** – инкапсуляция данных и поведения, с возможностью взаимодействия через публичный интерфейс + транспорт посредством отправки сообщений

2. **ООП как синтаксис** – механизмы ЯП, позволяющие удобнее придерживаться принципам и столпам ООП

3. **ООП как методология** – дизайн кода, использующий Объектную Инкапсуляции и как следствие этого использующий паттерны и принципы ООП

👋 Осознание

David Shekunts

ООП как архитектура – может быть замечательным решением

ООП как синтаксис – выбор каждого

А вот **ООП как методология**, которая пропагандирует Объектную Инкапсуляцию, вызывает много вопросов.

👋 Осознание

David Shekunts

Напишем небольшое ООП приложение

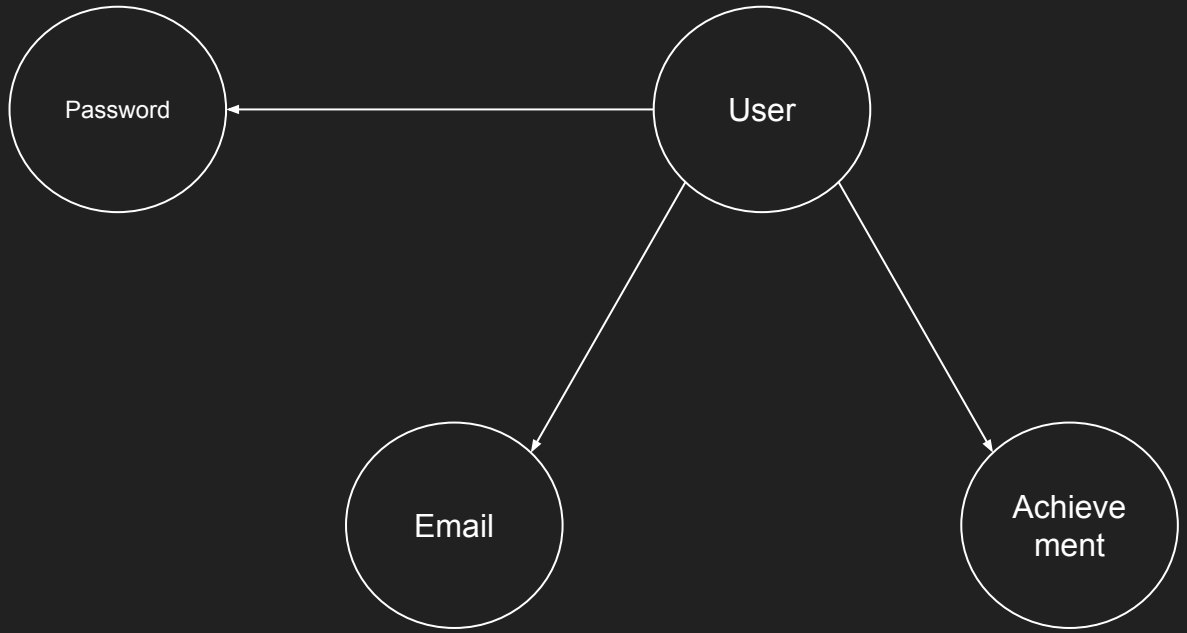
👋 Осознание

David Shekunts

Anemic Domain Model

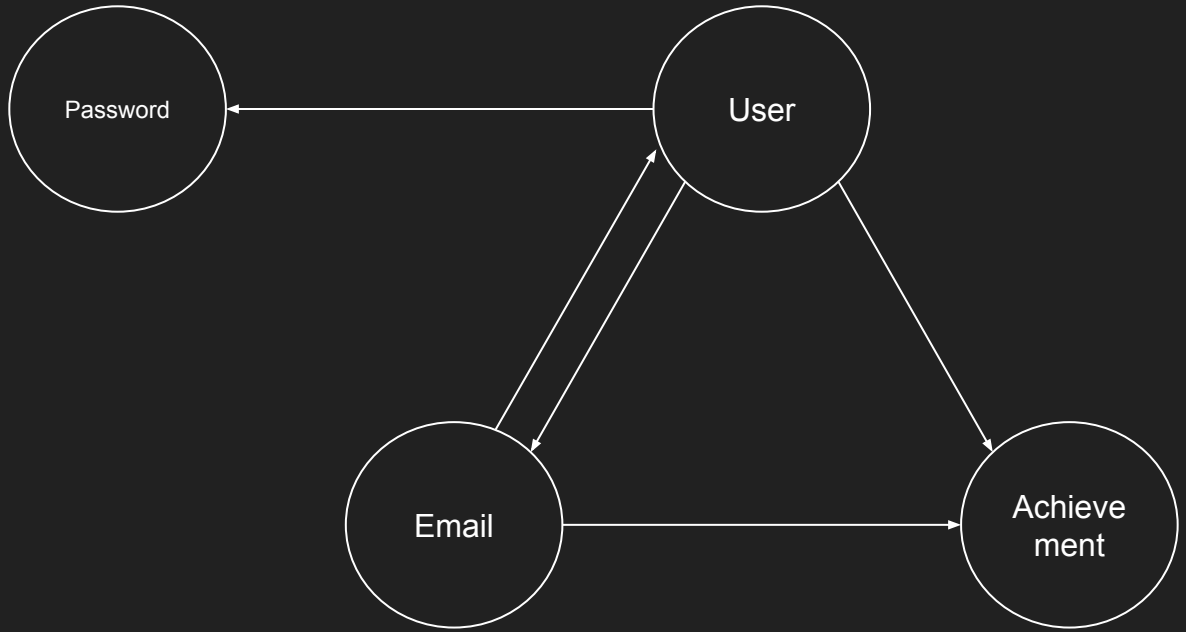
👋 Осознание

David Shekunts



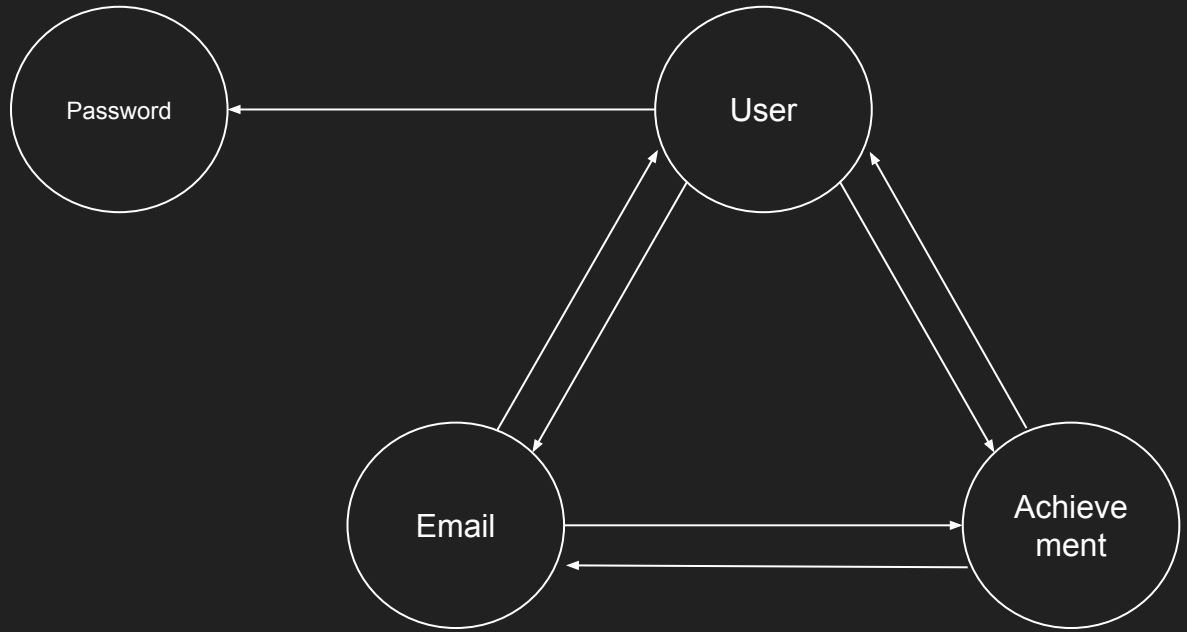
👋 Осознание

David Shekunts



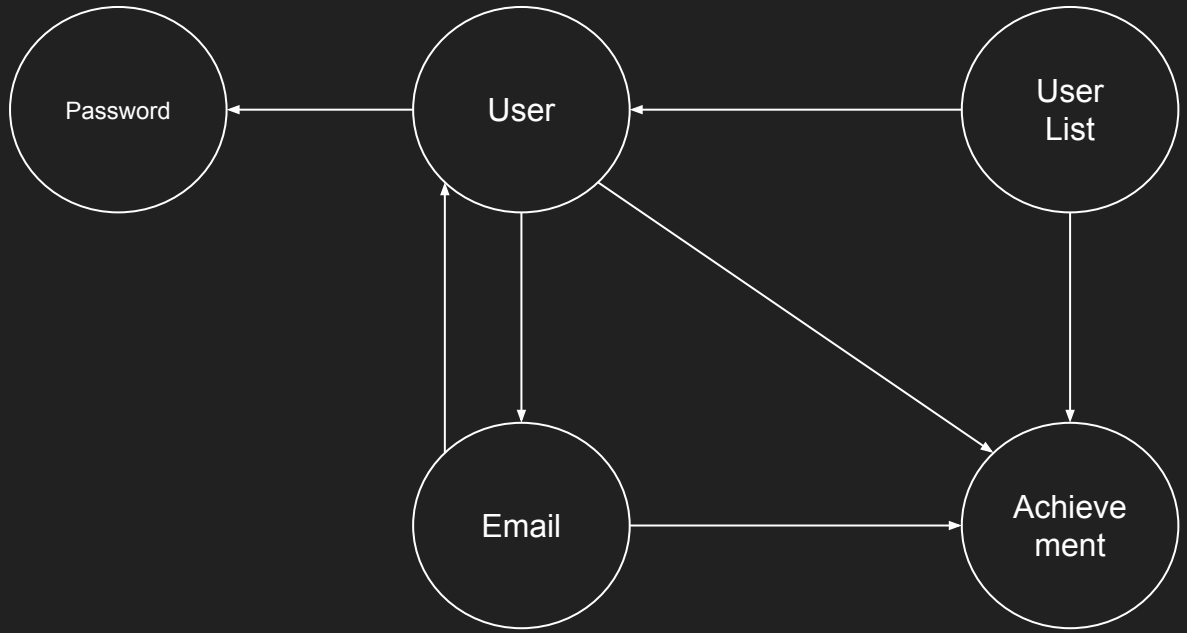
👋 Осознание

David Shekunts



👋 Осознание

David Shekunts



👋 Осознание

David Shekunts

💣 Корневая проблема ООП 💣

👋 Осознание

David Shekunts

2 стадии проекта

Создание

Когда мы можем все разработать “правильно”

Развитие

Когда разработанное становится “неправильным”

👋 Осознание

David Shekunts

При развитии программы для сохранения Объектной Инкапсуляции вам придется:

1. Создавать все больше и больше зависимостей в графе “объектов”
2. Расширять этот граф новыми абстракциями

👋 Осознание

David Shekunts

Увеличение зависимостей в графе “объектов”

рано или поздно приведет к Дед лупу

👋 Осознание

David Shekunts

Увеличение зависимостей в графе “объектов”
рано или поздно приведет к Дед лупу

->

Увеличению абстракция

👋 Осознание

David Shekunts

Увеличение зависимостей в графе “объектов”
рано или поздно приведет к Дед лупу

->

Увеличению абстракция

->

Рефакторингу + Увеличению Искусственную Сложности

👋 Осознание

David Shekunts

Мое определение:

1. **Натуральная**
2. **Искусственная**

👋 Осознание

David Shekunts

Мое определение

Натуральная

- Сложность алгоритма
- Устройство компьютера
- Устройство БД
- Геолокация

Искусственная

(абстракции)

- Архитектура
- Схема
- Методология
- Стайл-гайд

👋 Осознание

David Shekunts

Проблемы излишней искусственной сложности:

1. **Усложнение разработки**
2. **Уменьшение гибкости**
3. **Сложность дебагинга**

👋 Осознание

David Shekunts

ООП пропитан искусственной сложностью, потому что для решения задачи нам недостаточно “просто ее решить”, мы должны:

1. Выбрать подходящий “объект” или создать его
2. Распределить поведение между ответственными объектами
3. Вынести “лишнюю ответственность”
4. Раздробить текущие объекты и добавить новые

👋 Осознание

David Shekunts

💣 Корневая проблема ООП 💣

При развитии проекта для сохранения Объектной Инкапсуляции вам придется либо усложнять граф и сталкиваться с дед лупом, либо рефакторингу и увеличивать искусственную сложность, замедляя процесс разработки

👋 Осознание

David Shekunts

Еще пара проблем ООП:

1. Cross Cutting Concern
2. Частные методы в обобщенном пространстве
3. Degradation tendency

David Shekunts

🎭 Альтернативы 🎭

🎭 Альтернативы

David Shekunts

👱 А что с миксами? 👱

🎭 Альтернативы

David Shekunts

ООП + ФП / ПП = ☠️

1. По-прежнему подвержены проблеме абстракций
2. По-прежнему, частные методы в обобщенном пространстве
3. По-прежнему подвержены деградации
4. Кол-во точек изменения данных превращает дебагинг в боль

🎭 Альтернативы

И что остается?

🎭 Альтернативы

David Shekunts

Отказаться от ООП

🎭 Альтернативы

David Shekunts

🔬 ФП 🪬

👴🏿 ПП 👴🏿

🎭 Альтернативы

Чистый ФП в мультипарадигмальных языках – это чисто эстетика

🎭 Альтернативы

ФП – это stateless

🎭 Альтернативы

David Shekunts

```
1. const a = 10
2. sum(a, 2)
3. substract(a, 3)
```

🎭 Альтернативы

David Shekunts

```
1. const a = 10
2. sum(a, 2)
3. |-> substract(a, 3)
```

🎭 Альтернативы

David Shekunts

👴🏿 ПП 👴🏿

+

1. Не падает в абстракции
2. Простое и понятное
3. Легко работает в мультипарадигмальных ЯП

-

1. Устаревшие практики
2. Недостаток механик
3. Недостаток материалов

🎭 Альтернативы

ПП + ФП = ???

💡 Нахождение 💡

 Нахождение

David Shekunts

ПП + ФП = ???

1. Разделение Данных и Поведения
2. Композиция
3. Интерфейсный полиморфизм
4. Мутабельность

 Нахождение

ПП + ФП =

Функционально Ориентированное Программирование (ФОП)

👋 Осознание

David Shekunts

Вы уже писали по ФОП, если:

1. Использовали Anemic Domain Model
2. Писали на Go / Rust
3. Писали мутабельный ФП

David Shekunts

fop.davidshekunts.ru

👋 Итого

David Shekunts

Спасибо! С вами был Давид Шекунц и проект 🦾 IT-Качалка 💪

Tg – @it-kachalka