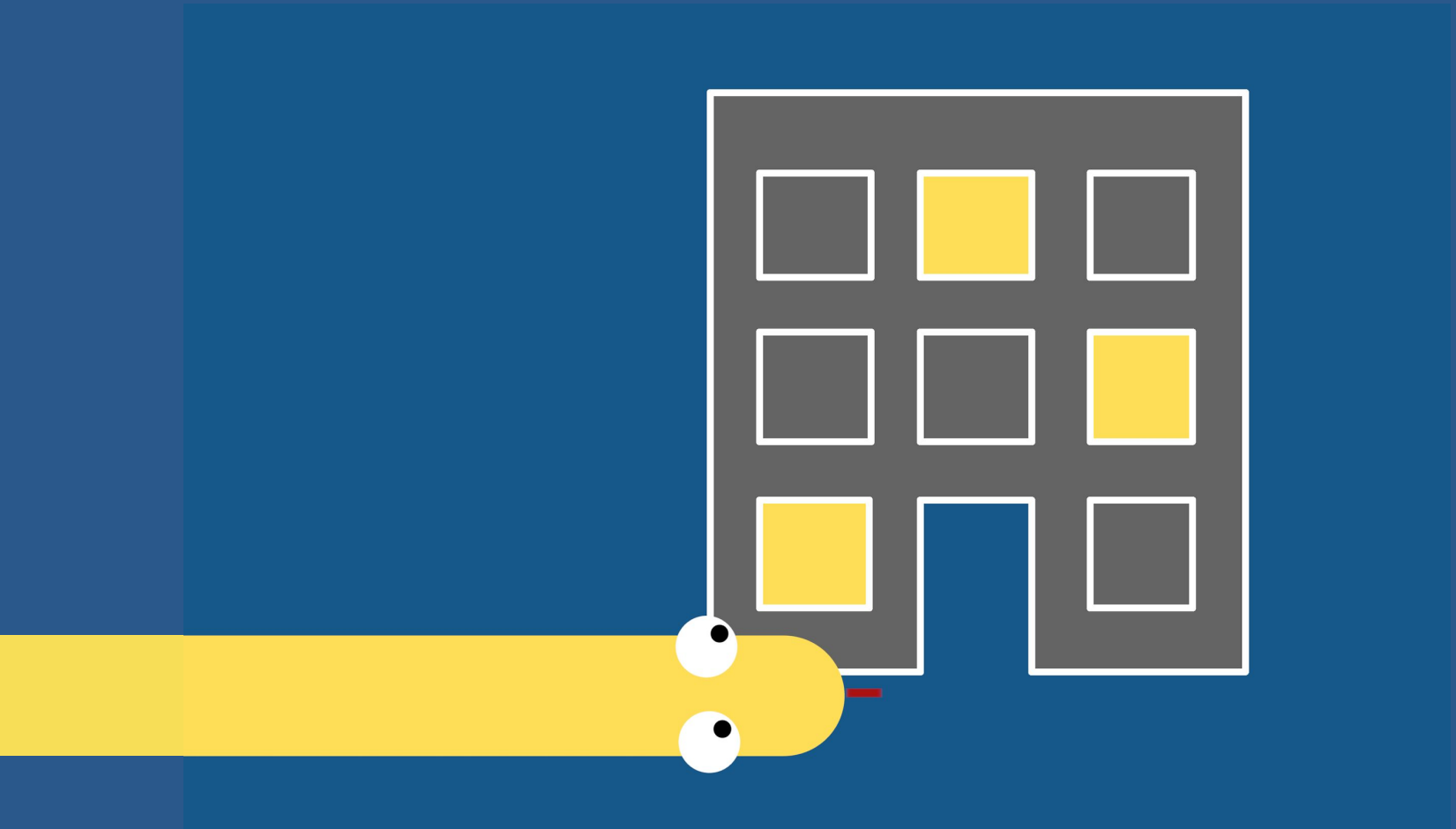
Python w krainie architektury



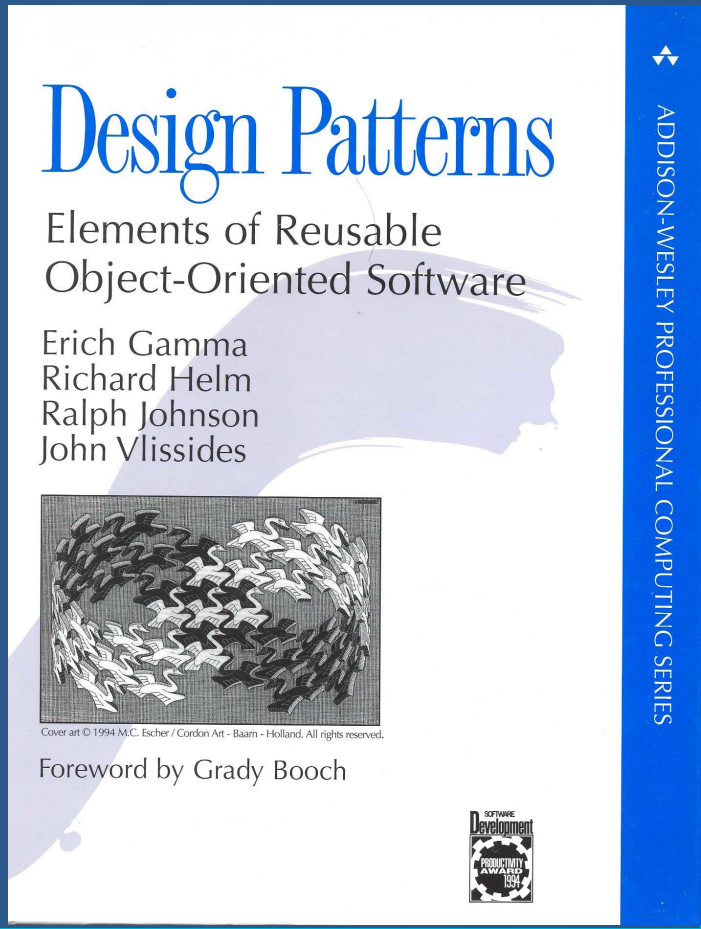
Czym jest architektura?

Czym jest architektura?

WIKIPEDIA
The Free Encyclopedia

Software architecture is about making fundamental structural choices that are costly to change once implemented. Software architecture choices include specific structural options from possibilities in the design of the software.

Czym jest architektura?



Czym jest architektura?

Shared understanding

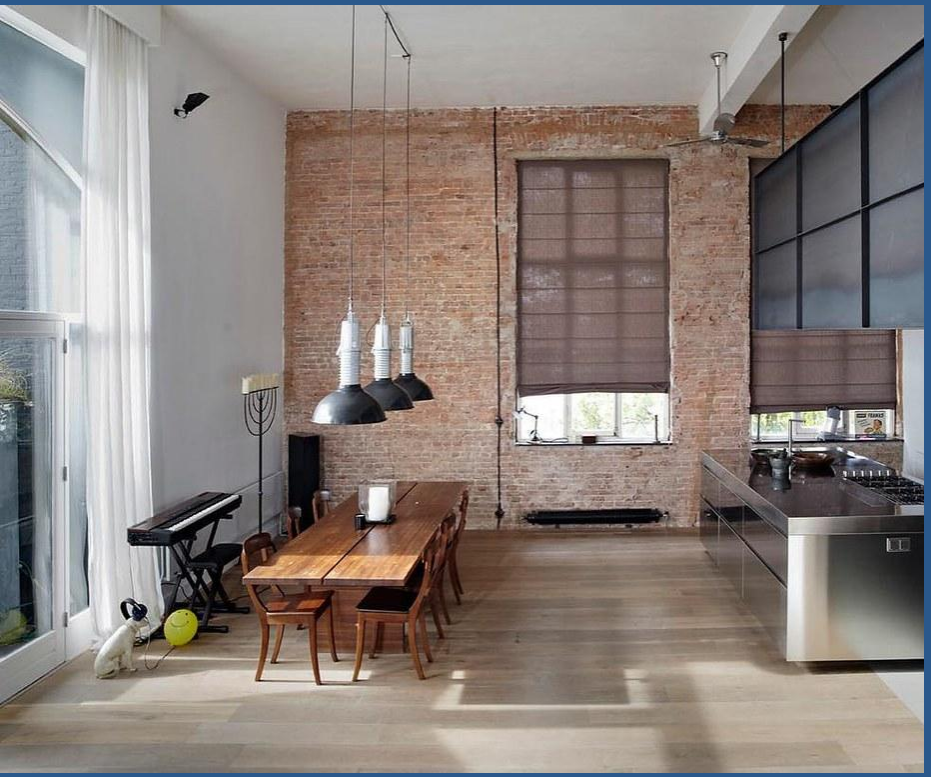
+

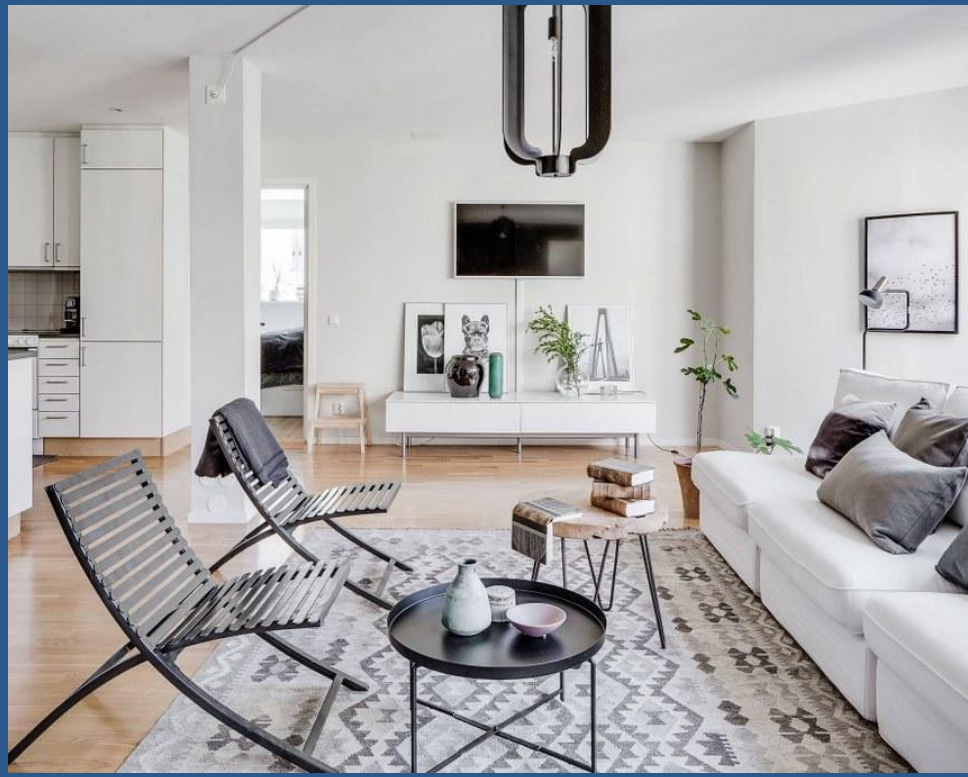
Hard to change

=

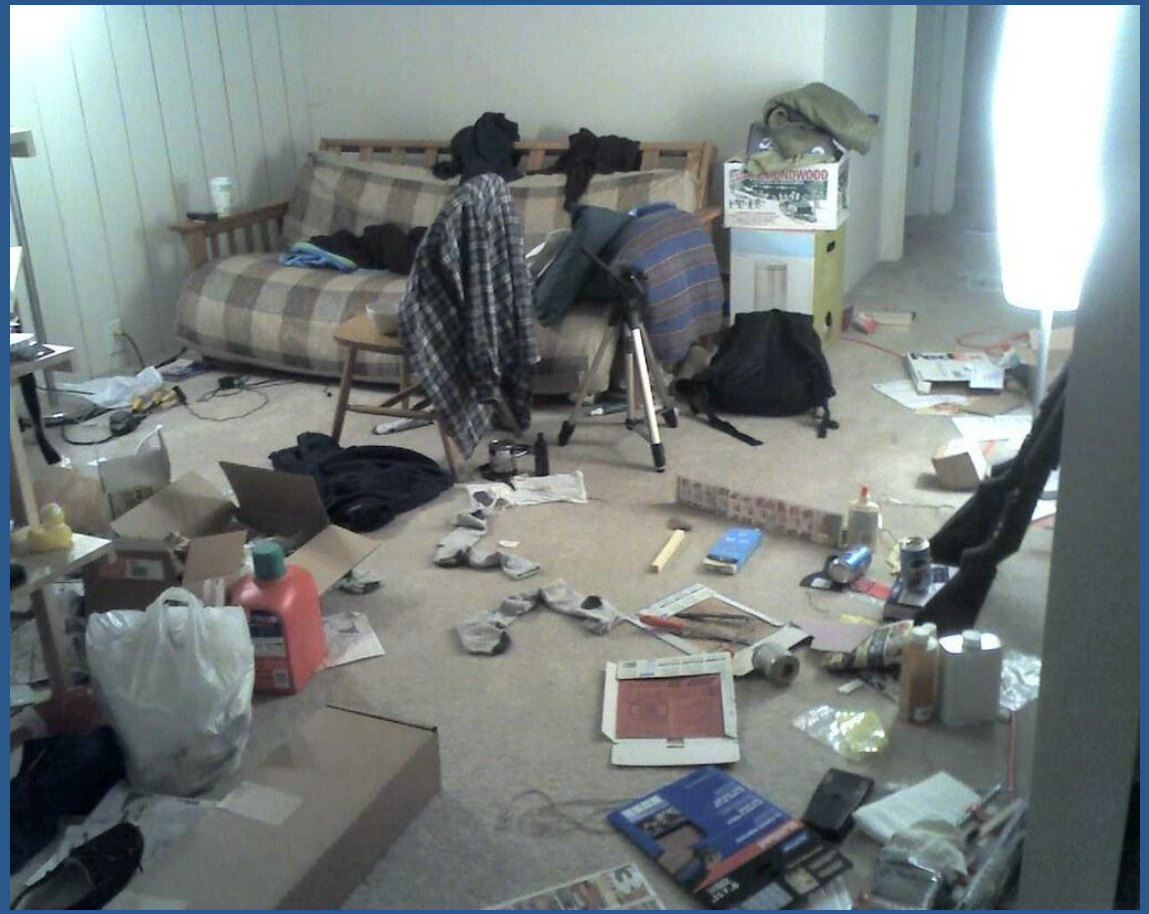
Important stuff

Styl oprogramowania

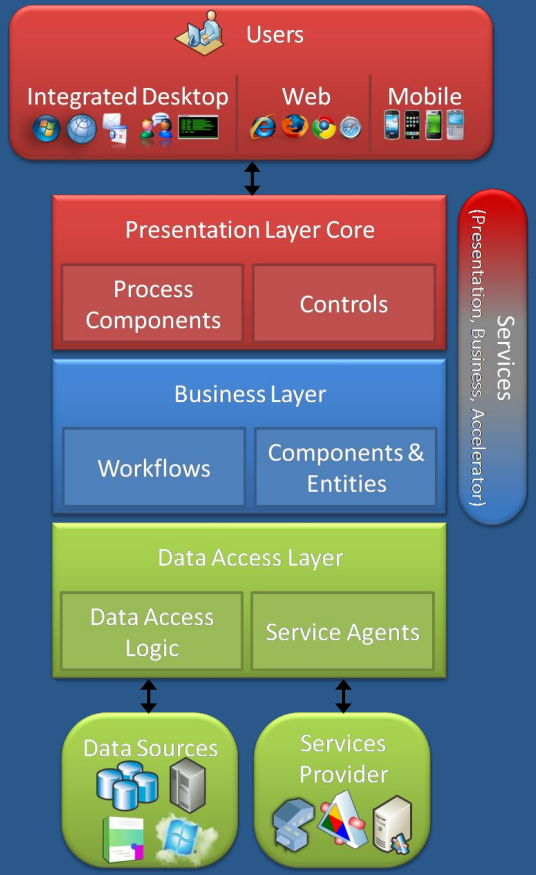




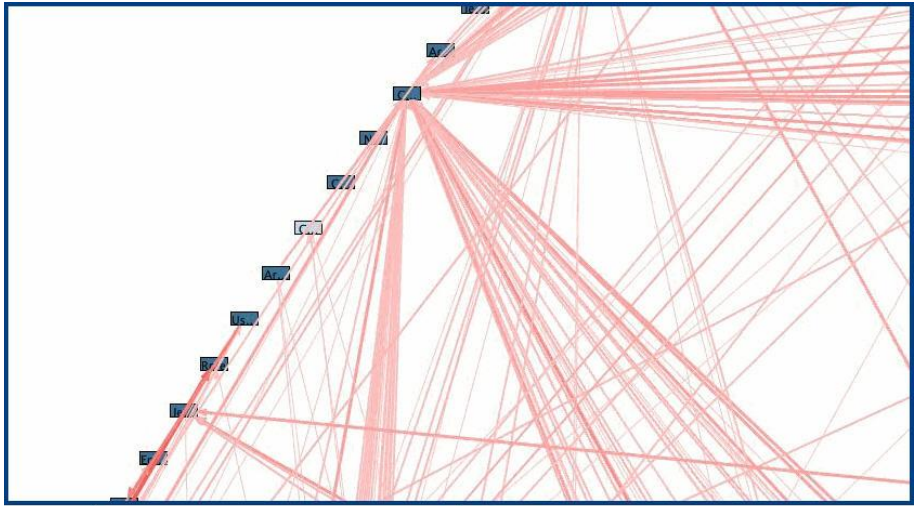
Styl oprogramowania



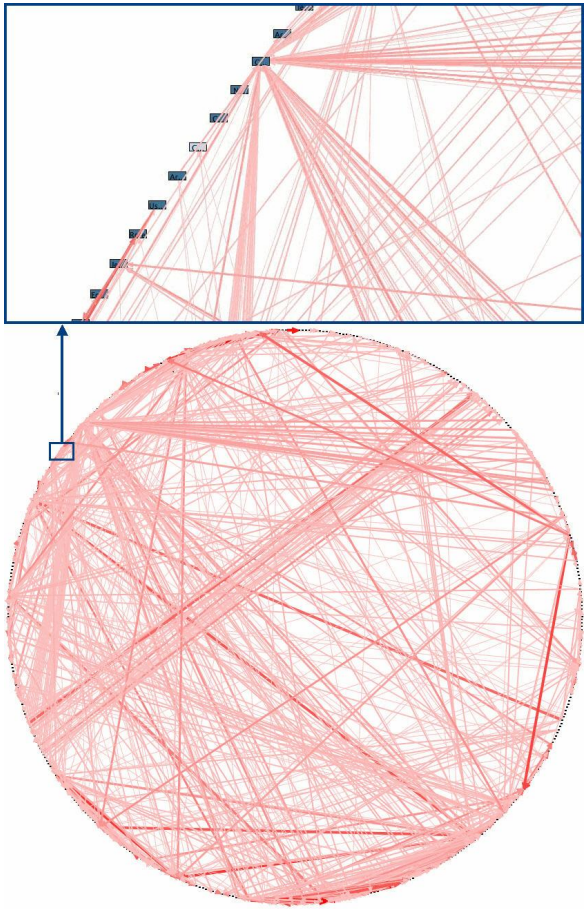
Styl oprogramowania



Styl oprogramowania



Styl oprogramowania



Czym jest architektura?

Architektura jest nieunikniona

Dobra architektura

Drivery architektoniczne

Funkcjonalność

Drivery architektoniczne

Funkcjonalność

Wymagania jakościowe

Drivery architektoniczne

Funkcjonalność

Wymagania jakościowe

Ograniczenia

Drivery architektoniczne

- Funkcjonalność
- Wymagania jakościowe
- Ograniczenia
- Konwencje, zasady

Czym jest architektura?

Architektura jest nieunikniona

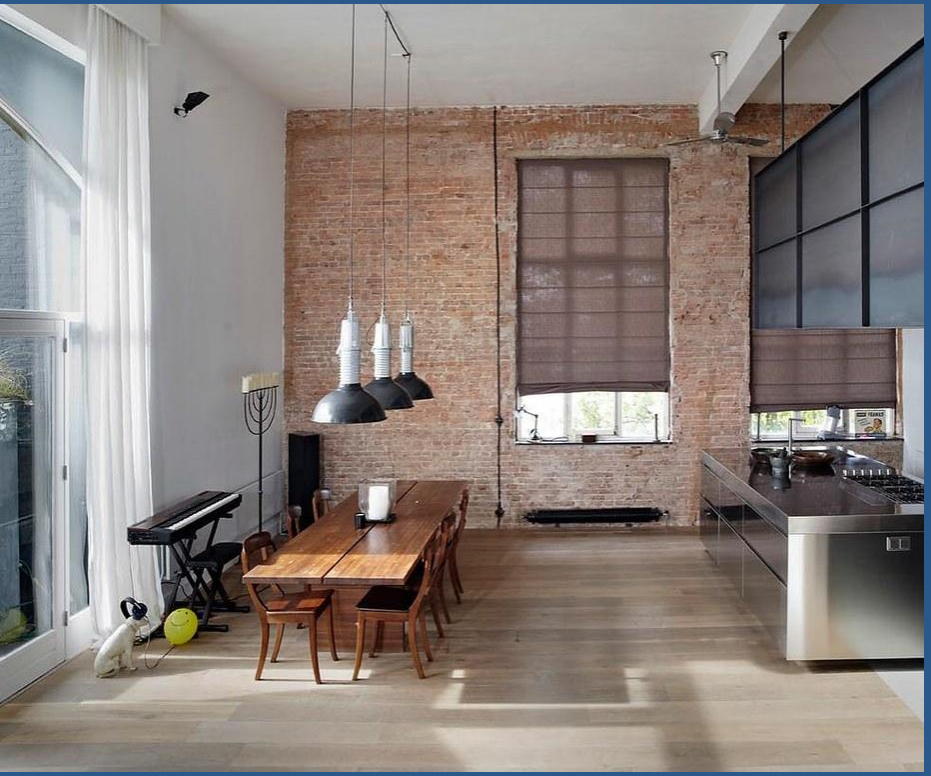
Czym jest architektura?

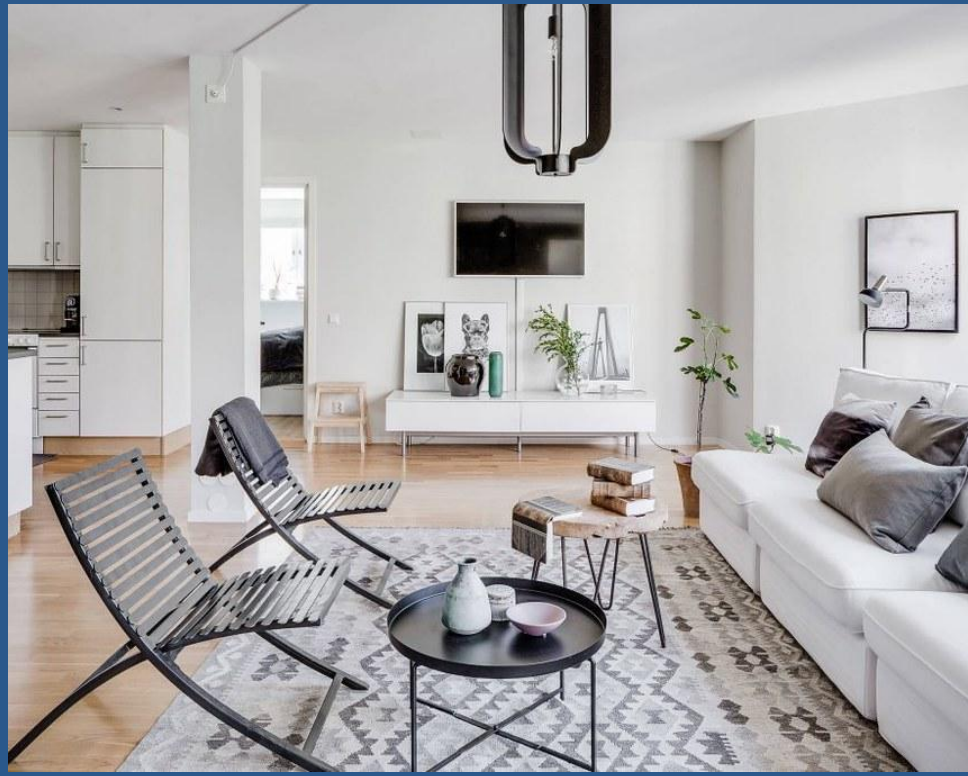
Architektura jest nieunikniona

Dobra architektura wspiera nasze cele (kontekst!)

Architektura na różnych poziomach

Architektura na różnych poziomach





Architektura aplikacyjna

Warstwowa

Architektura aplikacyjna

Warstwowa

Mixin/Magic Driven

Architektura aplikacyjna

Warstwowa

Mixin/Magic Driven

Clean/Hexagon

Architektura aplikacyjna

- Warstwowa
- Mixin/Magic Driven
- Clean/Hexagon
- Pipe and filters

Architektura aplikacyjna

- Warstwowa
- Mixin/Magic Driven
- Clean/Hexagon
- Pipe and filters
- Pluginowa

Czym jest architektura?

Architektura jest nieunikniona

Dobra architektura wspiera nasze cele (kontekst!)

Czym jest architektura?

Architektura jest nieunikniona

Dobra architektura wspiera nasze cele (kontekst!)

Są różne style architektoniczne

Czym jest architektura?

Architektura jest nieunikniona

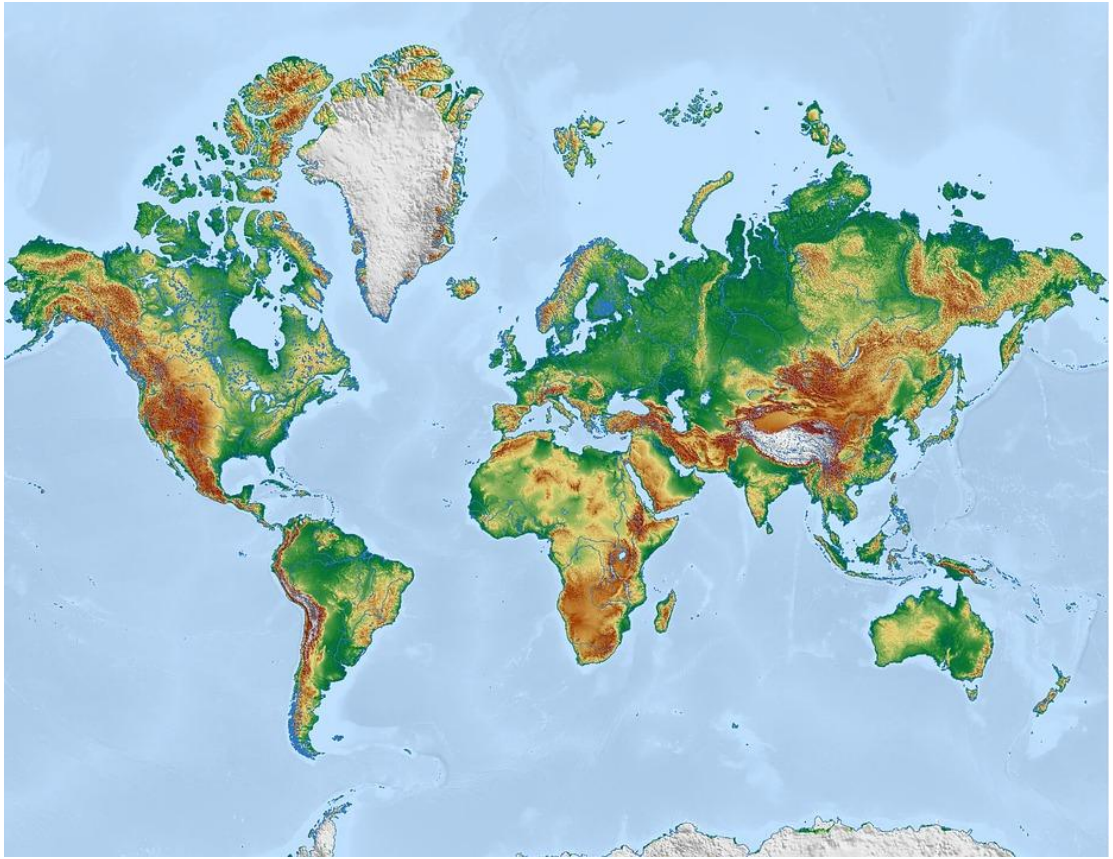
Dobra architektura wspiera nasze cele (kontekst!)

Są różne style architektoniczne

Dobór architektury nie musi być trudny

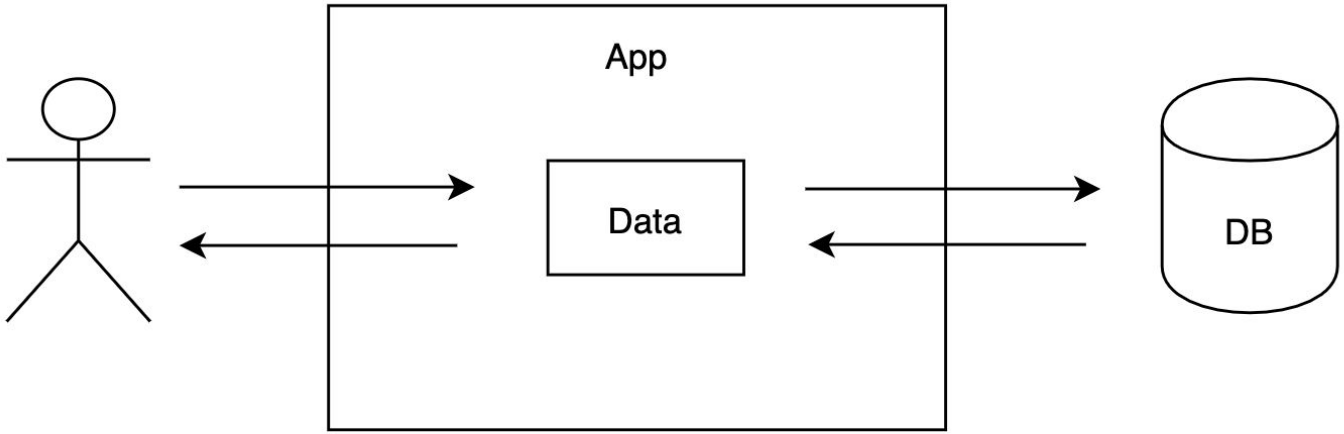
Model bogaty vs model anemiczny

Model bogaty vs model anemiczny



Model bogaty vs model anemiczny

Model anemiczny



Model anemiczny

```python
@api_view(["POST"])
def hold(request: Request) -> Response:
    insurance_identifier = request.GET.get("insurance_identifier")
    insurance = Insurance.objects.filter(identifier=insurance_identifier)

    if len(insurance.pauses) >= MAX_AVAILABLE_PAUSES:
        return response.bad_request('Pause limit exceeded')

    if insurance.status not in [Insurance.Status.ACTIVE or Insurance.Status.IN_GREY_PERIOD]:
        return response.bad_request('Wrong insurance status')

    now = datetime.datetime.now()
    # Transaction begin
    Pause.objects.create(insurance_identifier=insurance, begin_at=now)
    insurance.status = Insurance.Status.ON_HOLD
    insurance.save()
    # Transaction end

    return response.ok_no_content()
```

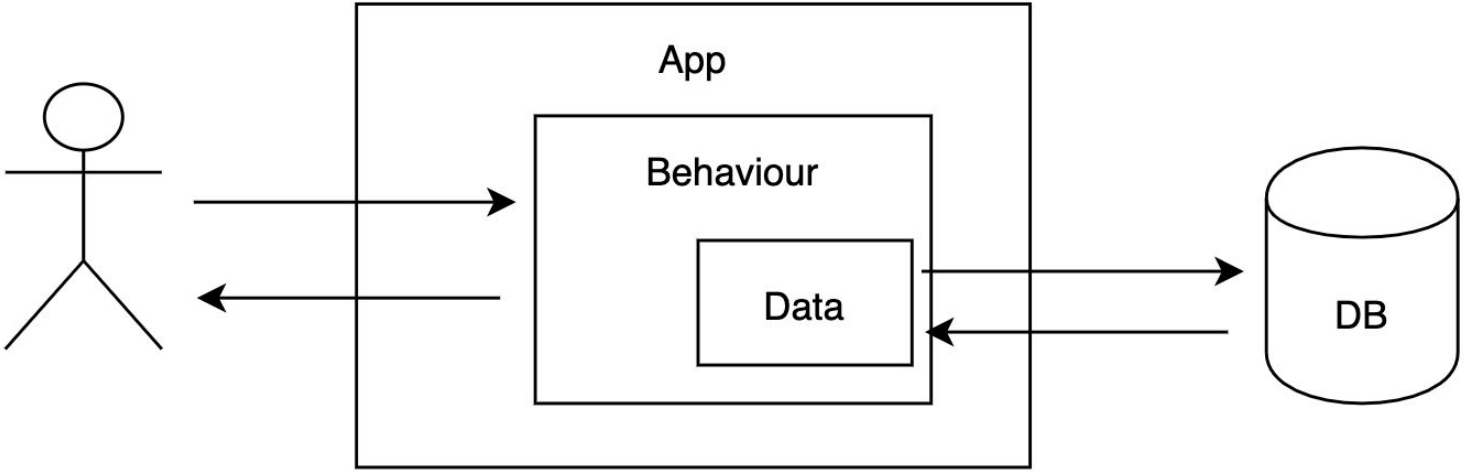
```python
class Pause(models.Model):
    insurance_identifier = models.ForeignKey(
        'Insurance', on_delete=models.CASCADE, related_name='pauses'
    )
    begin_at = models.DateTimeField()
    end_at = models.DateTimeField(null=True)


class Insurance(models.Model):
    class Status:
        ACTIVE = 'ACTIVE'
        ON_HOLD = 'ON_HOLD'
        IN_GREY_PERIOD = 'IN_GREY_PERIOD'
        INACTIVE = 'INACTIVE'

    insurance_status_choices = (
        (Status.ACTIVE, Status.ACTIVE),
        (Status.ON_HOLD, Status.ON_HOLD),
        (Status.IN_GREY_PERIOD, Status.IN_GREY_PERIOD),
        (Status.INACTIVE, Status.INACTIVE),
    )

    identifier = models.CharField(max_length=255)
    car_id = models.CharField(max_length=255)
    protection_end = models.DateTimeField()
    status = models.CharField(
        max_length=40, choices=insurance_status_choices
    )
```

Model bogaty



Model bogaty

```python
def resume(repo: InsuranceRepository, current_location: LocationProxy, identifier: str):
    insurance = repo.get(identifier)
    location = current_location(insurance.car_id)
    insurance.resume(location)
```

```python
@dataclass
class Pause:
    insurance_identifier: str
    begin_at: Datetime
    end_at: Optional[Datetime] = None

    def finish(self):
        if self.end_at:
            raise PauseAlreadyFinished()
        self.end_at = now


@dataclass
class Insurance:
    identifier: str
    car_id: str
    protection_end: Datetime
    status: InsuranceStatus
    pauses: List[Pause] = field(default_factory=list)

    def hold(self):
        if len(self.pauses) >= MAX_AVAILABLE_PAUSES:
            raise PauseLimitExceeded()

        if self.status not in [InsuranceStatus.ACTIVE or InsuranceStatus.IN_GREY_PERIOD]:
            raise WrongStateForAction()

        self.pauses.append(Pause(insurance_identifier=self.identifier, begin_at=now))
        self.status = InsuranceStatus.ON_HOLD

    def resume(self, current_location: Location):
        if self.status is not InsuranceStatus.ON_HOLD:
            raise WrongStateForAction()

        if not self._is_location_allowed(current_location):
```

Model anemiczny działa

```
class Product(models.Model):
    description = models.TextField()
    image = models.ImageField()
    category_description = models.TextField()
    price = models.IntegerField()
    activated = models.BooleanField()


class ProductSerializer(serializers.ModelSerializer):

    class Meta:
        model = Product
        fields = "__all__"


class ProductViewSet(ModelViewSet):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
```

Bardziej skomplikowany przykład

```
"""
Context:
    Car insurances with mobile application to hold/resume protection
    when working as driver. System need to store information about the insurance,
    its current status, track changes, subscription time, etc.
    We need to track pauses history (start/end)
    We want to be able to distinct between:
    - active (insurance is has not been hold and protection has not finished yet)
    - on hold (has been hold, no protection)
    - in grey period (tolerance for time to renew the insurance, resolve card issues, etc.)
    - inactive (insurance period and grey period have passed)

Use cases:
    1. Driver -> hold insurance when not at work
    Rules:
        Unable to hold when exceeded pause limit
        Unable to hold when already on hold
        Unable to hold when in inactive status
    2. Driver -> resume insurance when back at work
    Rules:
        Unable to resume when not currently on hold
        Unable to resume when outside the allowed location
    3. Driver -> pause for constant period
    Context:
        No matter how may times insurance has been hold driver can stop protection by using
        special pause - for constant time, without resume option.
        This time is fixed, however special pause can not overlap to a grey period.
        It should be included in overall on hold limit.
"""
```

Wybierz co potrzebujesz! (i wiedz dlaczego)

Wybierz co potrzebujesz! (i wiedz dlaczego)

PY_JAZZ

MIKOŁAJ_LEWANDOWSKI

/in/lewandowski-mikolaj

/PyJazz

PyJazz.pl

pythONpoczatek.pl