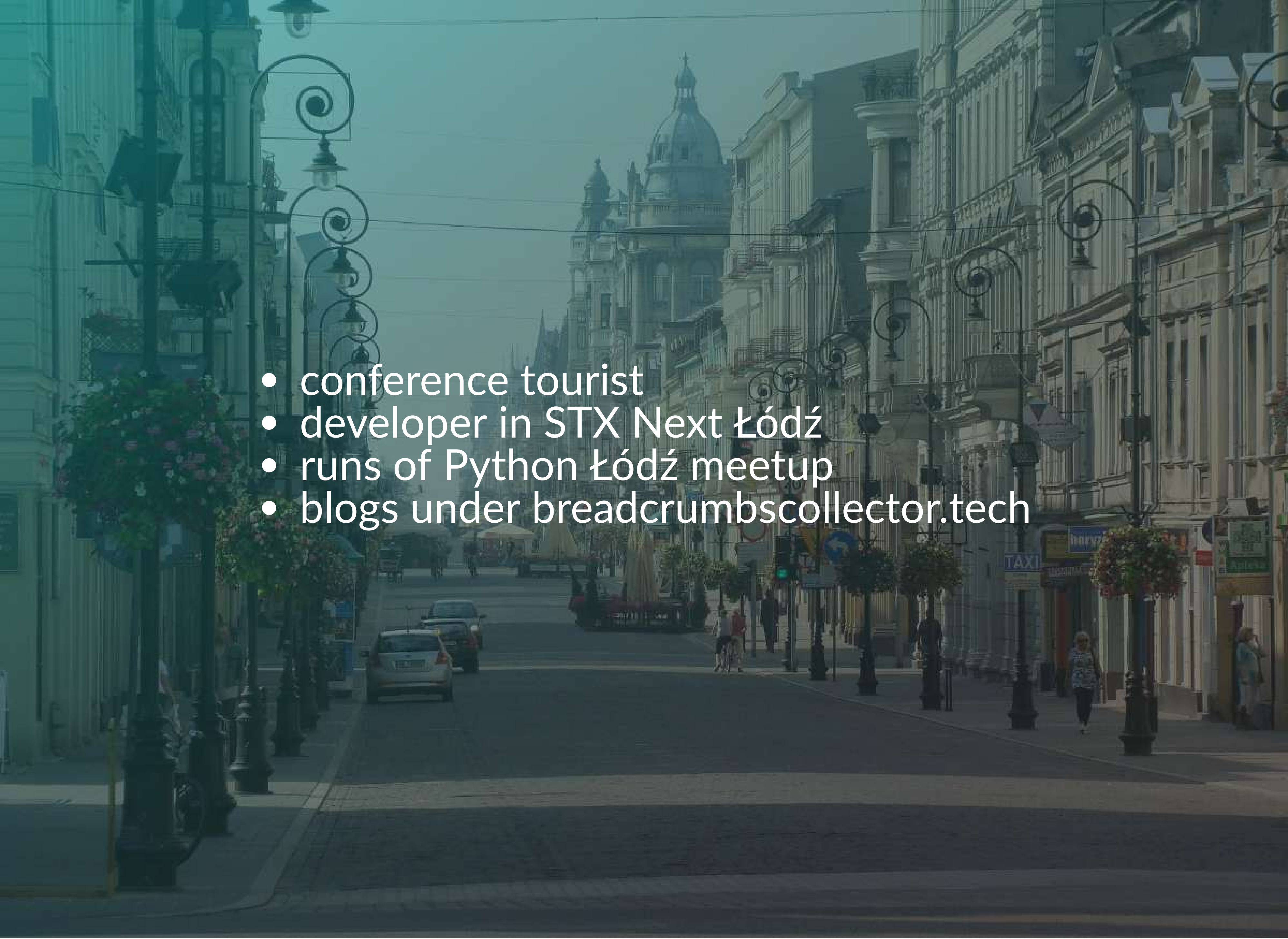
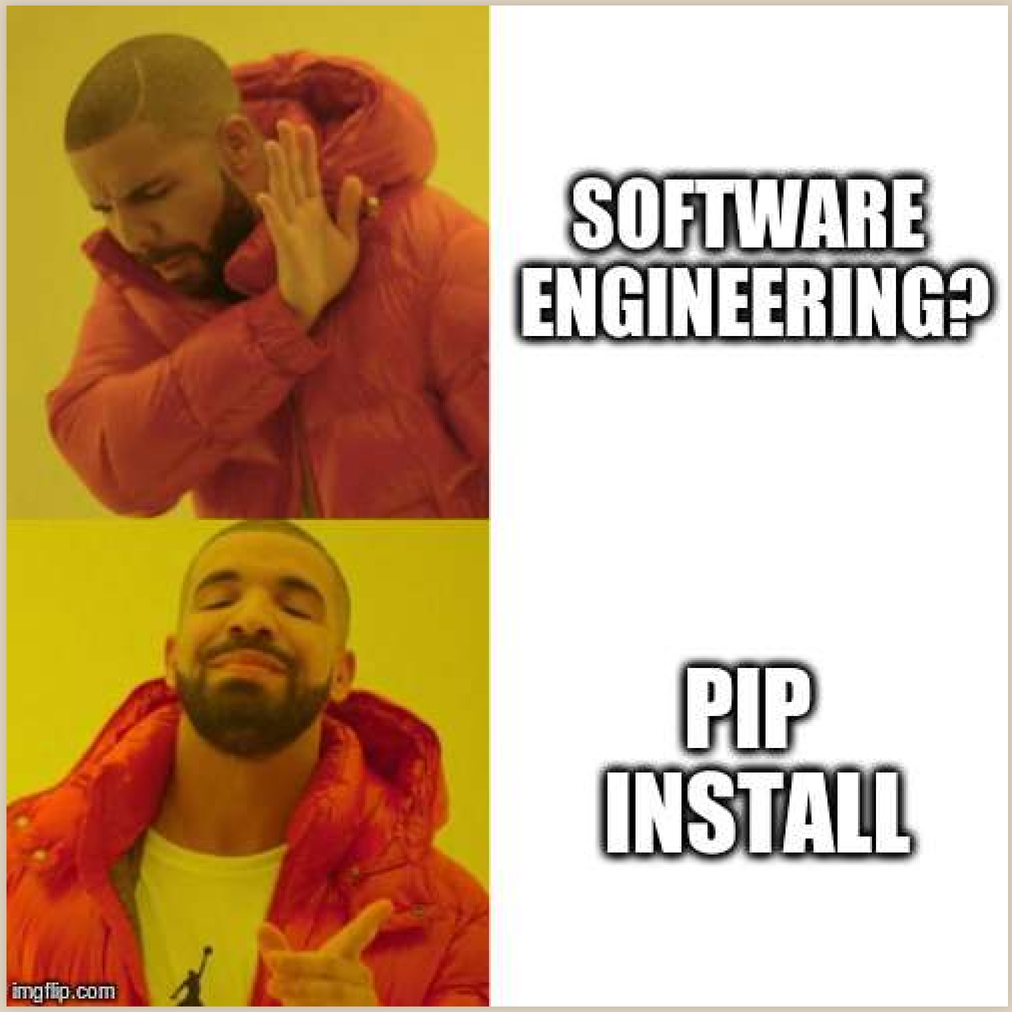
Clean Architecture

Sebastian Buczyński



- conference tourist
- developer in STX Next Łódź
- runs of Python Łódź meetup
- blogs under breadcrumbscollector.tech


SOFTWARE ENGINEERING?
PIP INSTALL
imgflip.com

complexity

Two types of complexity

- accidental
- essential

No Silver Bullet – Essence and Accident in Software Engineering

Clean Architecture

1. Independence of frameworks
2. Testability
3. Independence of UI or database

Clean Architecture

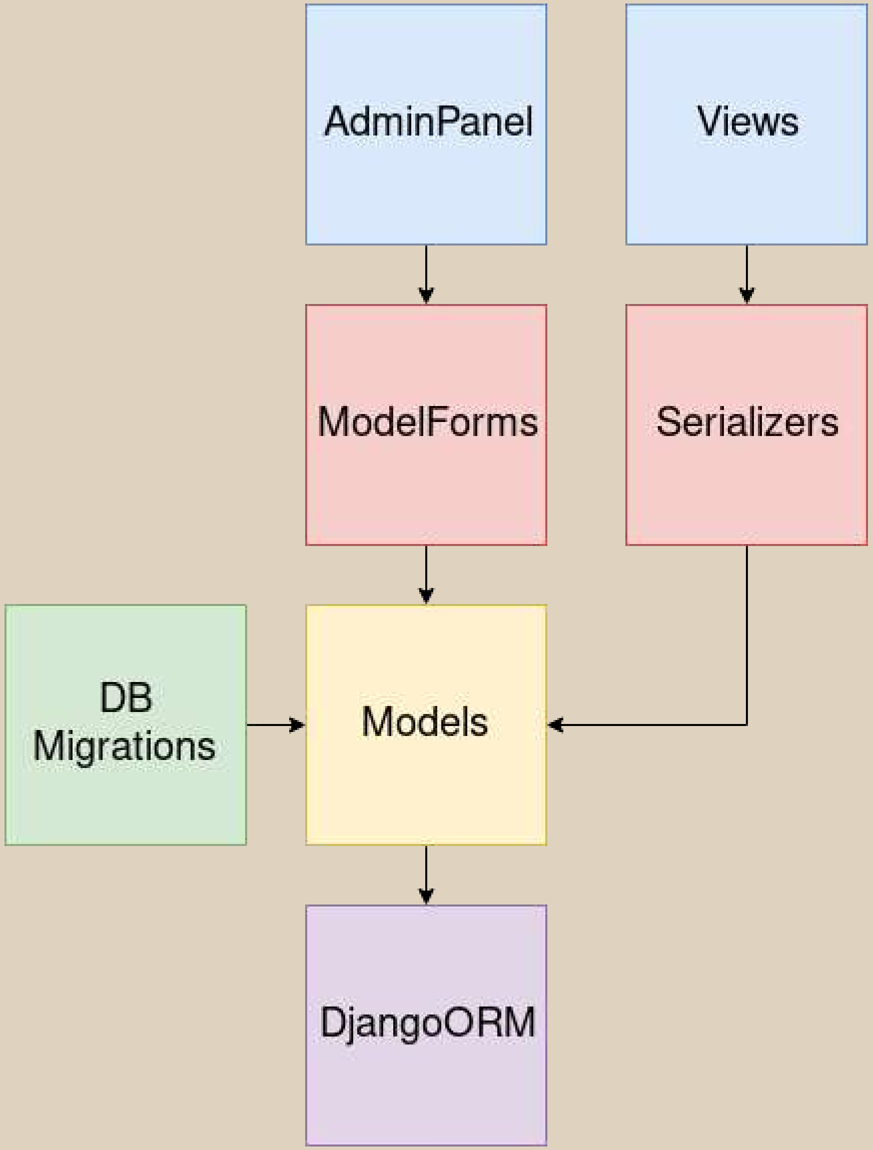
Separates complexity of your code

Project: Auctions online

User stories

- As a bidder I want to make a bid to win an auction
- As a bidder I want to be notified by e-mail when my bid is a winning one
- As an administrator I want to be able to withdraw a bid

Django + Rest Framework!


AdminPanel
Views
ModelForms
Serializers
DB Migrations
Models
DjangoORM

User stories -› code

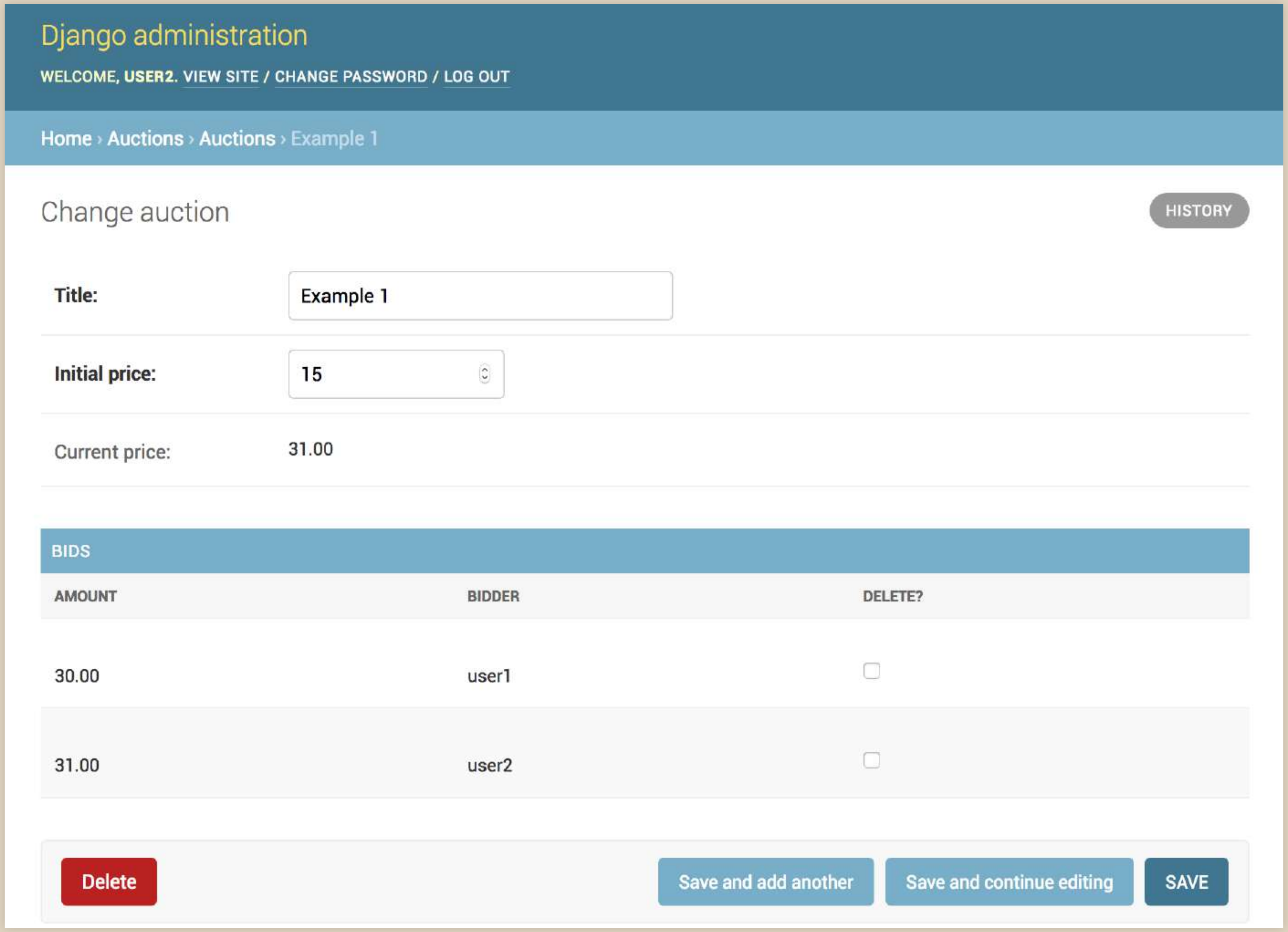
- As a bidder I want to make a bid to win an auction
- As a bidder I want to be notified by e-mail when my bid is a winning one
- As an administrator I want to be able to withdraw a bid

Models first

```
class Auction(models.Model):
    title = models.CharField(...)
    initial_price = models.DecimalField(...)
    current_price = models.DecimalField(...)


class Bid(models.Model):
    amount = models.DecimalField(...)
    bidder = models.ForeignKey(...)
    auction = models.ForeignKey(Auction, on_delete=PROTECT)
```

User stories

- ~~As a bidder I want to make a bid to win an auction~~ ✔
- ~~As a bidder I want to be notified by e-mail when my offer is a winning one~~ ✔
- As an administrator I want to be able to withdraw a bid


Django administration

WELCOME, **USER2**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Auctions › Auctions › Example 1

Change auction

HISTORY

Title:	Example 1
Initial price:	15
Current price:	31.00

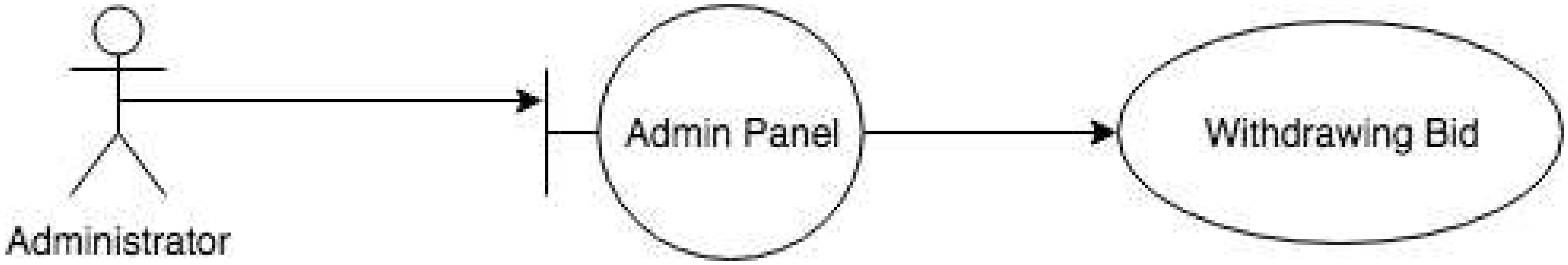
BIDS

AMOUNT	BIDDER	DELETE?
30.00	user1	☐
31.00	user2	☐

Delete | Save and add another | Save and continue editing | SAVE

```python
def save_related(self, request, form, formsets, *args, **kwargs):
    ids_of_deleted_bids = self._get_ids_of_deleted_bids(formsets)
    bids_to_withdraw = Bid.objects.filter(
        pk__in=ids_of_deleted_bids)

    auction = form.instance
    old_winners = set(auction.winners)
    auction.withdraw_bids(bids_to_withdraw)
    new_winners = set(auction.winners)

    self._notify_winners(new_winners - old_winners)

    super().save_related(request, _form, formsets, *args, **kwarg
def save_related(self, request, form, formsets, *args, **kwargs):
    ids_of_deleted_bids = self._get_ids_of_deleted_bids(formsets)
    bids_to_withdraw = Bid.objects.filter(
        pk__in=ids_of_deleted_bids)

    auction = form.instance
    old_winners = set(auction.winners)
    auction.withdraw_bids(bids_to_withdraw)
    new_winners = set(auction.winners)

    self._notify_winners(new_winners - old_winners)

    super().save_related(request, _form, formsets, *args, **kwarg
```

```python
def save_related(self, request, form, formsets, *args, **kwargs):
    ids_of_deleted_bids = self._get_ids_of_deleted_bids(formsets)
    bids_to_withdraw = Bid.objects.filter(
        pk__in=ids_of_deleted_bids)

    auction = form.instance
    old_winners = set(auction.winners)
    auction.withdraw_bids(bids_to_withdraw)
    new_winners = set(auction.winners)

    self._notify_winners(new_winners - old_winners)

    super().save_related(request, _form, formsets, *args, **kwarg
```


Administrator
Admin Panel
Withdrawing Bid

Clean Arch - building block #1

```
class WithdrawingBid:
    def withdraw_bids(self, auction_id, bids_ids):
        auction = Auction.objects.get(pk=auction_id)
        bids_to_withdraw = Bid.objects.filter(
            pk__in=ids_of_deleted_bids)

        old_winners = set(auction.winners)
        auction.withdraw_bids(bids_to_withdraw)
        new_winners = set(auction.winners)

        self._notify_winners(new_winners - old_winners)
```

UseCase OR Interactor

UseCase - Orchestrates a particular process

What about tests?!

Business logic is coupled with a framework, so are tests...

Testing through views

```
from django.test import TestCase

class LoginTestCase(TestCase):

    def test_login(self):
        User.objects.create(...)

        response = self.client.get('/dashboard/')

        self.assertRedirects(response, '/accounts/login/')
```

FOLLOWED
OCK
FOLLOWED

How much time wasted, exactly?

https://breadcrumbscollector.tech/is-your-test-suite-wasting-your-time/

How a textbook example looks like?

```
class MyTest(unittest.TestCase):
    def test_add(self):
        expected = 7

        actual = add(3, 4)

        self.assertEqual(actual, expected)
```

No side effects and dependencies makes code easier to test

PURE FUNCTION

Getting rid of dependencies: find them

```
class WithdrawingBidUseCase:
    def withdraw_bids(self, auction_id, bids_ids):
        auction = Auction.objects.get(pk=auction_id)
        bids_to_withdraw = Bid.objects.filter(
            pk__in=ids_of_deleted_bids)

        old_winners = set(auction.winners)
        auction.withdraw_bids(bids_to_withdraw)
        new_winners = set(auction.winners)

        self._notify_winners(new_winners - old_winners)
```

Getting rid of dependencies: hide them

```python
class WithdrawingBidUseCase:
    def withdraw_bids(self, auction_id, bids_ids):
        auction = self.auctions_repository.get(auction_id)
        bids = self.bids_repository.get_by_ids(bids_ids)

        old_winners = set(auction.winners)
        auction.withdraw_bids(bids)
        new_winners = set(auction.winners)

        self.auctions_repository.save(auction)
        for bid in bids:
            self.bids_repository.save(bid)

        self._notify_winners(new_winners - old_winners)
```

Getting rid of dependencies: hide them

```
class WithdrawingBidUseCase:
    def withdraw_bids(self, auction_id, bids_ids):
        auction = self.auctions_repository.get(auction_id)
        bids = self.bids_repository.get_by_ids(bids_ids)

        old_winners = set(auction.winners)
        auction.withdraw_bids(bids)
        new_winners = set(auction.winners)

        self.auctions_repository.save(auction)
        for bid in bids:
            self.bids_repository.save(bid)

        self._notify_winners(new_winners - old_winners)
```

Clean Arch - building block #2

```
class AuctionsRepo(metaclass=ABCMeta):

    @abstractmethod
    def get(self, auction_id):
        pass

    @abstractmethod
    def save(self, auction):
        pass
```

Interface / Port

PayPal

Square

DUE

Authorize.Net®

wepay

adyen

stripe

amazon pay

2CHECKOUT

Clean Arch - building block #2

```
class AuctionsRepo(metaclass=ABCMeta):

    @abstractmethod
    def get(self, auction_id):
        pass

    @abstractmethod
    def save(self, auction):
        pass
```

Interface / Port

Clean Arch - building block #3

```python
class DjangoAuctionsRepo(AuctionsRepo):

    def get(self, auction_id):
        return Auction.objects.get(pk=auction_id)
```

Interface Adapter / Adapter

Combine together

```python
class WithdrawingBidUseCase:
    def __init__(self, auctions_repository: AuctionsRepo):
        self.auctions_repository = auctions_repository
```

```python
django_adapter = DjangoAuctionsRepo()
withdrawing_bid_uc = WithdrawingBidUseCase(django_adapter)
```

Dependency Injection

```python
import inject

def configure_inject(binder: inject.Binder):
    binder.bind(AuctionsRepo, DjangoAuctionsRepo())

inject.configure_once(configure_inject)
```

```python
class WithdrawingBidUseCase:

    auctions_repo: AuctionsRepo = inject.attr(AuctionsRepo)
```

Benefits from another layer

- It is easier to reason about logic
- It is possible to write TRUE unit tests
- Work can be parallelized
- **Decision making can be deferred**
- OOP done right

Our logic is still coupled to a database!

```python
class WithdrawingBidUseCase:
    def withdraw_bids(self, auction_id, bids_ids):
        auction = self.auctions_repository.get(auction_id)
        bids = self.bids_repository.get_by_ids(bids_ids)

        old_winners = set(auction.winners)
        auction.withdraw_bids(bids)
        new_winners = set(auction.winners)

        self.auctions_repository.save(auction)
        for bid in bids:
            self.bids_repository.save(bid)

        self._notify_winners(new_winners - old_winners)
```

Clean Arch - building block #0

```python
class Auction:
    def __init__(self, id: int, title: str, bids: List[Bid]):
        self.id = id
        self.title = title
        self.bids = bids

    def withdraw_bids(self, bids: List[Bid]):
        ...

    def make_a_bid(self, bid: Bid):
        ...

    @property
    def winners(self):
        ...
```

Entity

Clean Arch - building block #3

```python
class DjangoAuctionsRepo(AuctionsRepo):
    def get(self, auction_id: int) -> Auction:
        auction_model = AuctionModel.objects.prefetch_related(
            'bids'
        ).get(pk=auction_id)

        bids = [
            self._bid_from_model(bid_model)
            for bid_model in auction_model.bids.all()
        ]

        return Auction(
            auction_model.id,
            auction_model.title,
            bids
        )
```

Interface Adapter / Adapter

Entity vs model #1

```
auction = Auction(id=1, title='Super auction', bids=[])

auction.bids.append(Bid())  ✗

auction.make_a_bid(Bid())  ✓
```

Entity = data & **rules** - adhere to **Tell, don't ask** principle

Entity vs model #2

Entity can represent graph of objects

Clean Arch building blocks altogether

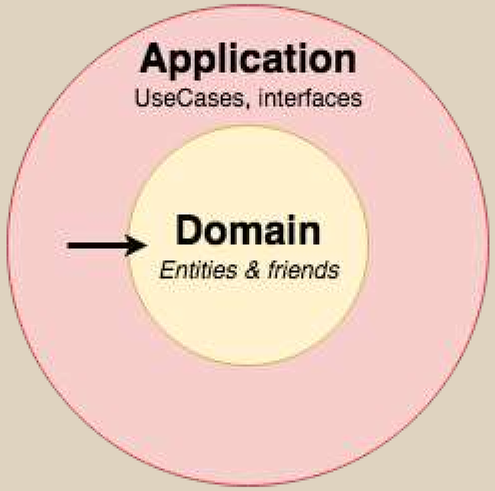
- Entity
- Interface / Port
- Interface Adapter / Adapter
- Use Case / Interactor
- Presenter*
- + space for more

*see exemplary project

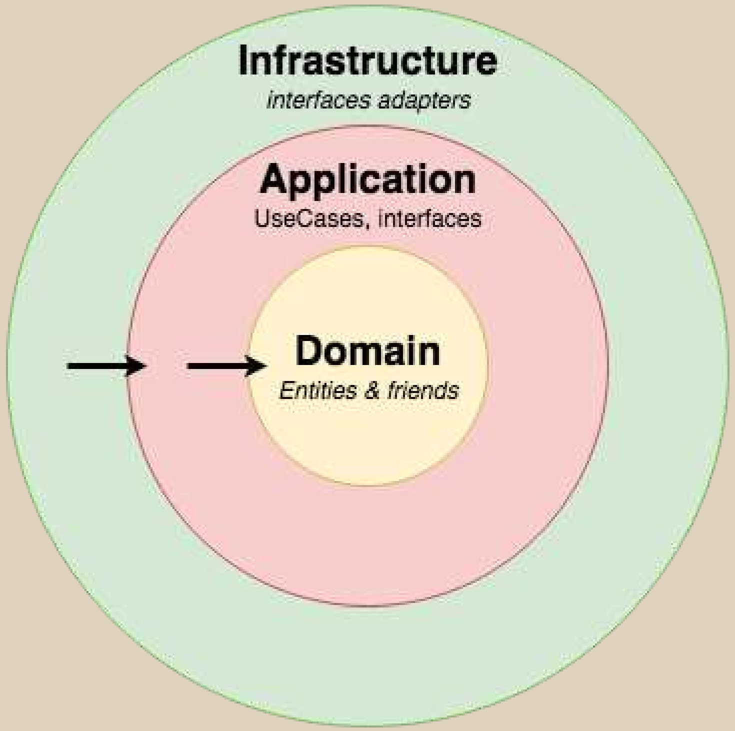
Clean Arch building blocks altogether

Domain
Entities & friends

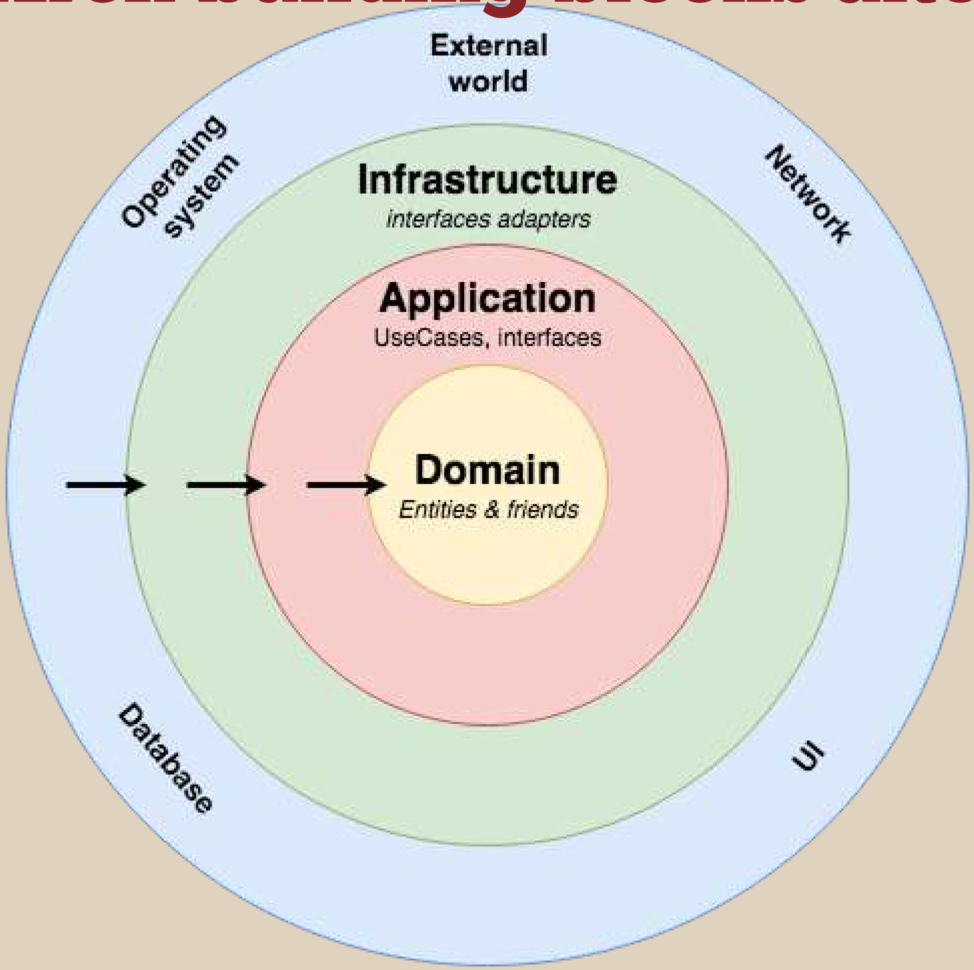
Clean Arch building blocks altogether



Clean Arch building blocks altogether

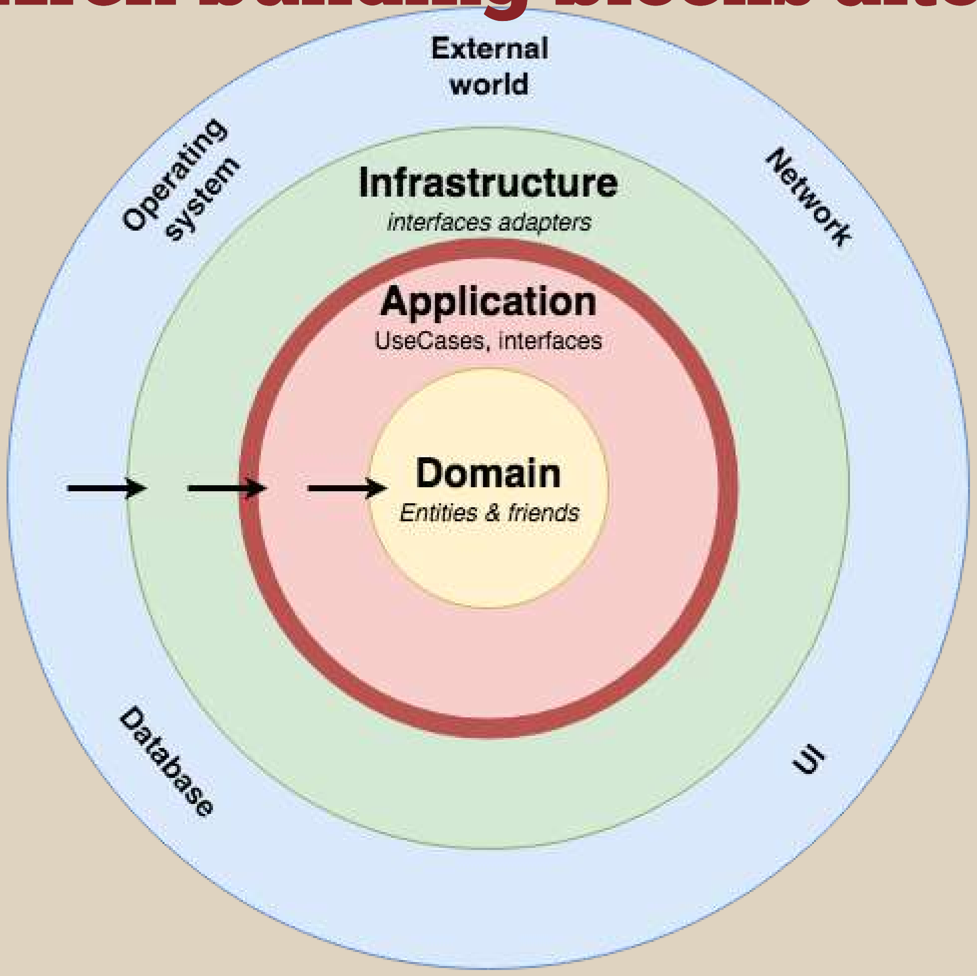


Clean Arch building blocks altogether



You MUST NOT use/import anything from a layer above!

Clean Arch building blocks altogether



Boundary

What to be careful of?

non-idiomatic framework use

Word on frameworks

- ✓ Pyramid
- ✓ Flask
- ✗ Django*

*if you like pain

more code (type hints help)

copying data between objects

validation?

DRF serializers, colander, marshmallow, typechecking

value objects

```
money = Decimal('10.00')  # meh
```

value objects

```
money = Money('10.00012')  # raises ValueError
money = Money('10.12$')  # yay!
```

overengineering

When it pays off?

lots of cases - testability

Testing entities

```python
def test_should_use_initial_price_as_current_price_when_no_bids()
    auction = create_auction()

    assert auction.current_price == auction.initial_price


def test_should_return_highest_bid_amount_for_current_price():
    auction = create_auction(bids=[
        Bid(id=1, bidder_id=1, amount=Decimal('20')),
        Bid(id=2, bidder_id=2, amount=Decimal('15')),
    ])

    assert auction.current_price == Decimal('20')
```

Testing use cases

```python
def test_saves_auction(
        auctions_repo_mock: Mock,
        auction_mock: Mock,
        input_dto: PlacingBidInputDto
) -> None:
    PlacingBidUseCase().execute(input_dto)

    auctions_repo_mock.save.assert_called_once_with(auction_mock)
```

deferring decision making - stay lean

complicated domain

Futher reading

https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

Clean Architecture: A Craftsman's Guide to Software Structure and Design

Clean Architecture Python (web) apps - Przemek Lewandowski

Software architecture chronicles - blog posts series

Boundaries - Gary Bernhardt

Exemplary project in PHP (blog post)

Exemplary project in PHP (repo)

Exemplary project in C# (repo)

Exemplary project in Python (repo)

Czysta Architektura: Jak stworzyć testowalny i elastyczny kod (justjoin.it)

‹shameless plug›

I'm writing a book!

cleanarchitecture.io

‹/shameless plug›

That's all, folks!

Questions?

cleanarchitecture.io/talk