Refactoring legacy Django app using OOP

whoami

- Principal Software Developer @ Webinterpret
- Author of Implementing The Clean Architecture
- Blogger @ breadcrumbscollector.tech
- Advocate of Software Engineering in Python world

Refactoring

improving the design of existing code

Refactoring is a journey

```
def share_daily_random_quote(order):
    quote_for_today = random.choice(QUOTES)

    client = WebClient(
        token=os.environ['SLACK_BOT_TOKEN']
    )
    client.chat_postMessage(
        channel='#random', text=quote_for_today
    )
```

```
def share_daily_random_quote(order):
    quote_for_today = random.choice(QUOTES)


def post_to_slack(quote):
    client = WebClient(
        token=os.environ['SLACK_BOT_TOKEN']
    )
    client.chat_postMessage(
        channel='#random', text=quote_for_today
    )
```

Extract function

```
def share_daily_random_quote(order):
    quote_for_today = random.choice(QUOTES)
    post_to_slack(quote_for_today)


def post_to_slack(quote):
    client = WebClient(
        token=os.environ['SLACK_BOT_TOKEN']
    )
    client.chat_postMessage(
        channel='#random', text=quote_for_today
    )
```

Extract function

Rename Method

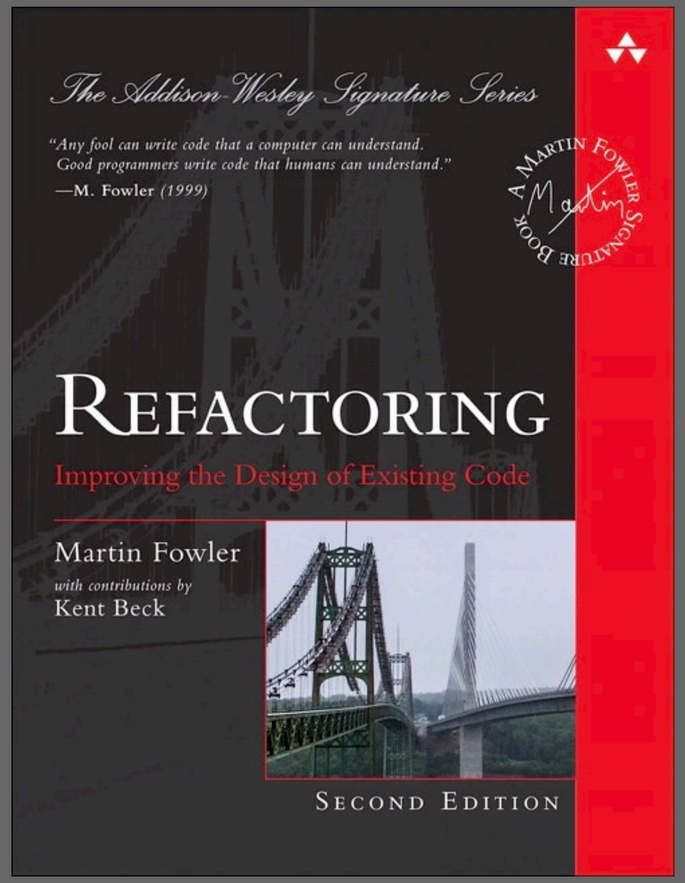
Extract Superclass

Extract Class

Introduce Assertion

...

Extract function
Rename Method
Extract Superclass
Extract Class
Introduce Assertion
...



or refactoring.guru


No Refactoring
without tests

The best moment to refactor?

Right before implementing a new feature


legacy
django
APP

```python
class BookingsViewSet(CreateModelMixin, RetrieveModelMixin, GenericViewSet):
    queryset = Booking.objects.all()
    serializer_class = BookingSerializer
```

```python
class BookingSerializer(HyperlinkedModelSerializer):

    amusement_park_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=AmusementParkPreBooking.objects.all(),
    )
    museum_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=MuseumPreBooking.objects.all(),
    )
    restaurant_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=RestaurantPreBooking.objects.all()
    )
    payment_card_token = CharField(write_only=True)
    ...
```

```
class BookingSerializer(HyperlinkedModelSerializer):

    amusement_park_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=AmusementParkPreBooking.objects.all(),
    )
    museum_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=MuseumPreBooking.objects.all(),
    )
    restaurant_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=RestaurantPreBooking.objects.all()
    )
    payment_card_token = CharField(write_only=True)
    ...
```

```python
class BookingSerializer(HyperlinkedModelSerializer):

    amusement_park_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=AmusementParkPreBooking.objects.all(),
    )
    museum_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=MuseumPreBooking.objects.all(),
    )
    restaurant_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=RestaurantPreBooking.objects.all()
    )
    payment_card_token = CharField(write_only=True)
    ...
```

```
class BookingSerializer(HyperlinkedModelSerializer):

    amusement_park_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=AmusementParkPreBooking.objects.all(),
    )
    museum_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=MuseumPreBooking.objects.all(),
    )
    restaurant_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=RestaurantPreBooking.objects.all()
    )
    payment_card_token = CharField(write_only=True)
    ...
```

```
class BookingSerializer(HyperlinkedModelSerializer):

    amusement_park_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=AmusementParkPreBooking.objects.all(),
    )
    museum_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=MuseumPreBooking.objects.all(),
    )
    restaurant_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=RestaurantPreBooking.objects.all()
    )
    payment_card_token = CharField(write_only=True)
    ...
```

```python
class BookingSerializer(HyperlinkedModelSerializer):

    amusement_park_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=AmusementParkPreBooking.objects.all(),
    )
    museum_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=MuseumPreBooking.objects.all(),
    )
    restaurant_prebookings = PrimaryKeyRelatedField(
        many=True,
        queryset=RestaurantPreBooking.objects.all()
    )
    payment_card_token = CharField(write_only=True)
    ...
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
def create(self, validated_data):
    payment_card_token = validated_data.pop('payment_card_token')
    booking: Booking = super().create(validated_data)

    if not self.validate_payment_card_with_zero_auth(payment_card_token):
        booking.fail_payment()
        return booking

    if booking.pay_now_total.amount:
        with transaction.atomic():
            booking.authorize_payment_at_creation(
                card_token=payment_card_token
            )
            if booking.status == booking.Status.FAILED_PAYMENT:
                transaction.on_commit(
                    lambda: booking.send_email_about_failure()
                )
                return booking

    return self.finish_booking(booking)
```

```
    @staticmethod
    def finish_booking(booking: Booking):
        command = FinishBookingCommand()
        try:
            command.finish_booking(booking)
        except MadeUpApiProviderError:
            if booking.pay_now_total.amount:
                booking.cancel_payment()
            booking.send_email_about_failure('booking failed')
            return booking

        if booking.pay_now_total.amount:
            booking.capture_payment()

        booking.sync_with_crm()

        return booking
```

```
    @staticmethod
    def finish_booking(booking: Booking):
        command = FinishBookingCommand()
        try:
            command.finish_booking(booking)
        except MadeUpApiProviderError:
            if booking.pay_now_total.amount:
                booking.cancel_payment()
            booking.send_email_about_failure('booking failed')
            return booking

        if booking.pay_now_total.amount:
            booking.capture_payment()

        booking.sync_with_crm()

        return booking
```

```
    @staticmethod
    def finish_booking(booking: Booking):
        command = FinishBookingCommand()
        try:
            command.finish_booking(booking)
        except MadeUpApiProviderError:
            if booking.pay_now_total.amount:
                booking.cancel_payment()
            booking.send_email_about_failure('booking failed')
            return booking

        if booking.pay_now_total.amount:
            booking.capture_payment()

        booking.sync_with_crm()

        return booking
```

```
@staticmethod
def finish_booking(booking: Booking):
    command = FinishBookingCommand()
    try:
        command.finish_booking(booking)
    except MadeUpApiProviderError:
        if booking.pay_now_total.amount:
            booking.cancel_payment()
        booking.send_email_about_failure('booking failed')
        return booking

    if booking.pay_now_total.amount:
        booking.capture_payment()

    booking.sync_with_crm()

    return booking
```

```
@staticmethod
def finish_booking(booking: Booking):
    command = FinishBookingCommand()
    try:
        command.finish_booking(booking)
    except MadeUpApiProviderError:
        if booking.pay_now_total.amount:
            booking.cancel_payment()
        booking.send_email_about_failure('booking failed')
        return booking

    if booking.pay_now_total.amount:
        booking.capture_payment()

    booking.sync_with_crm()

    return booking
```

```
    @staticmethod
    def finish_booking(booking: Booking):
        command = FinishBookingCommand()
        try:
            command.finish_booking(booking)
        except MadeUpApiProviderError:
            if booking.pay_now_total.amount:
                booking.cancel_payment()
            booking.send_email_about_failure('booking failed')
            return booking

        if booking.pay_now_total.amount:
            booking.capture_payment()

        booking.sync_with_crm()

        return booking
```

```
@staticmethod
def finish_booking(booking: Booking):
    command = FinishBookingCommand()
    try:
        command.finish_booking(booking)
    except MadeUpApiProviderError:
        if booking.pay_now_total.amount:
            booking.cancel_payment()
        booking.send_email_about_failure('booking failed')
        return booking

    if booking.pay_now_total.amount:
        booking.capture_payment()

    booking.sync_with_crm()

    return booking
```

@staticmethod considered a code smell

```python
class FinishBookingCommand:
    def finish_booking(self, booking: Booking) -> None:
        ...

        try:
            booking_response = self.madeup_booking_client.book_at_once()
        except madeup_booking.MadeUpApiProviderError:
            logger.error('Oh no ...  Anyway')
            raise

        self.update_booking(booking, booking_response)

    @staticmethod
    def update_booking(booking: Booking, booking_response: dict) -> None:
        booking.reference = booking_response['ReferenceNumber']
        booking.save()
```

```python
class FinishBookingCommand:
    def finish_booking(self, booking: Booking) -> None:
        ...

        try:
            booking_response = self.madeup_booking_client.book_at_once()
        except madeup_booking.MadeUpApiProviderError:
            logger.error('Oh no ... Anyway')
            raise

        self.update_booking(booking, booking_response)

    @staticmethod
    def update_booking(booking: Booking, booking_response: dict) -> None:
        booking.reference = booking_response['ReferenceNumber']
        booking.save()
```

```python
class FinishBookingCommand:
    def finish_booking(self, booking: Booking) -> None:
        ...

        try:
            booking_response = self.madeup_booking_client.book_at_once()
        except madeup_booking.MadeUpApiProviderError:
            logger.error('Oh no ... Anyway')
            raise

        self.update_booking(booking, booking_response)

    @staticmethod
    def update_booking(booking: Booking, booking_response: dict) -> None:
        booking.reference = booking_response['ReferenceNumber']
        booking.save()
```

```python
class FinishBookingCommand:
    def finish_booking(self, booking: Booking) -> None:
        ...

        try:
            booking_response = self.madeup_booking_client.book_at_once()
        except madeup_booking.MadeUpApiProviderError:
            logger.error('Oh no ... Anyway')
            raise

        self.update_booking(booking, booking_response)

    @staticmethod
    def update_booking(booking: Booking, booking_response: dict) -> None:
        booking.reference = booking_response['ReferenceNumber']
        booking.save()
```

```python
def validate_payment_card_with_zero_auth(
        self, payment_card_token: str
    ) -> bool:
        if waffle.switch_is_active('use_new_payment'):
            try:
                madeup_payment.Charge.create( ... )
            except madeup_payment.PaymentFailed:
                return False

            return True
        else:
            client = old_payment.Client()
            response = client.authorize( ... )
            if response.status == old_payment.Status.SUCCESS:
                return True
            else:
                return False
```

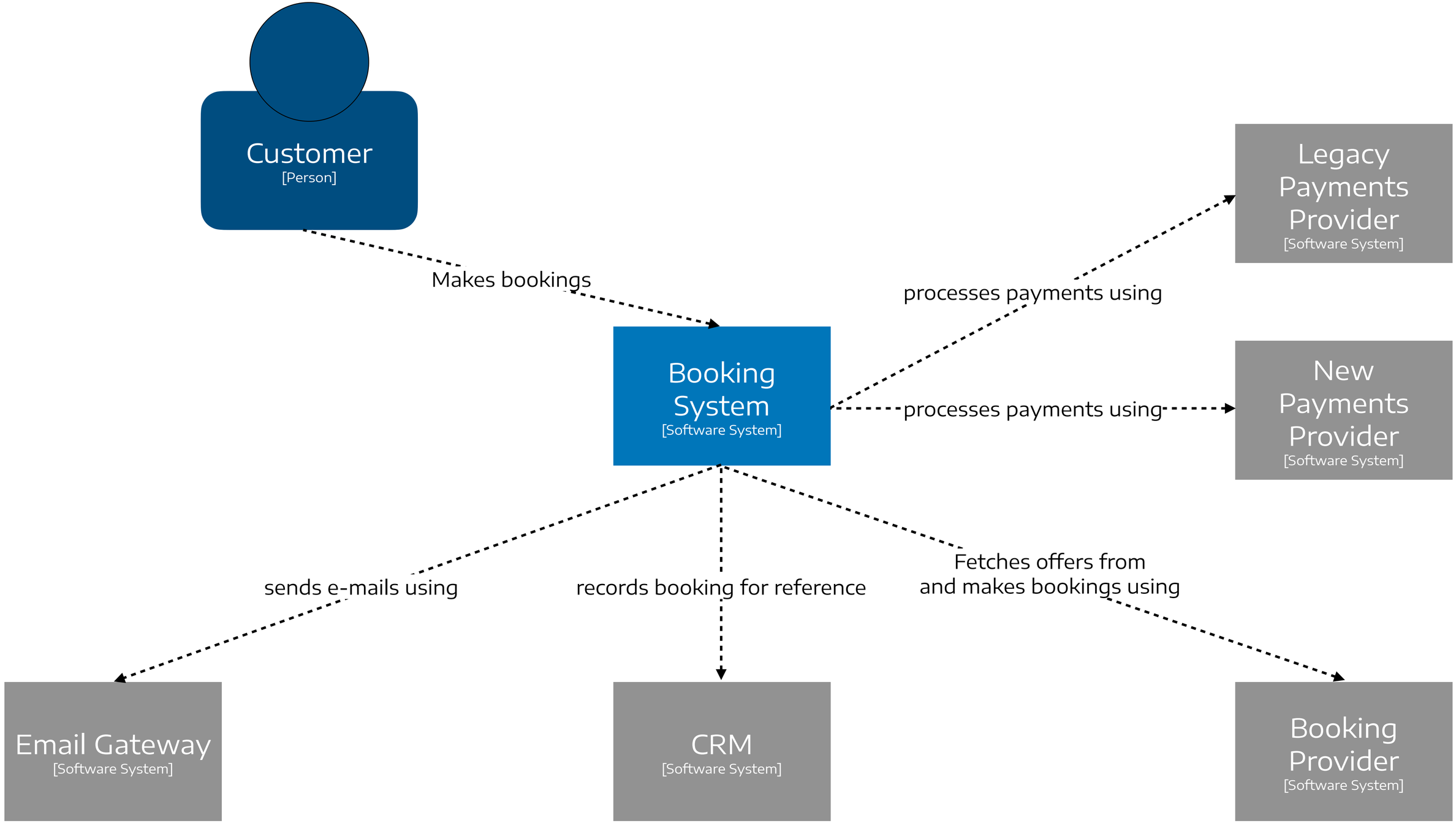
```
def validate_payment_card_with_zero_auth(
    self, payment_card_token: str
) → bool:
    if waffle.switch_is_active('use_new_payment'):
        try:
            madeup_payment.Charge.create( ... )
        except madeup_payment.PaymentFailed:
            return False

        return True
    else:
        client = old_payment.Client()
        response = client.authorize( ... )
        if response.status == old_payment.Status.SUCCESS:
            return True
        else:
            return False
```

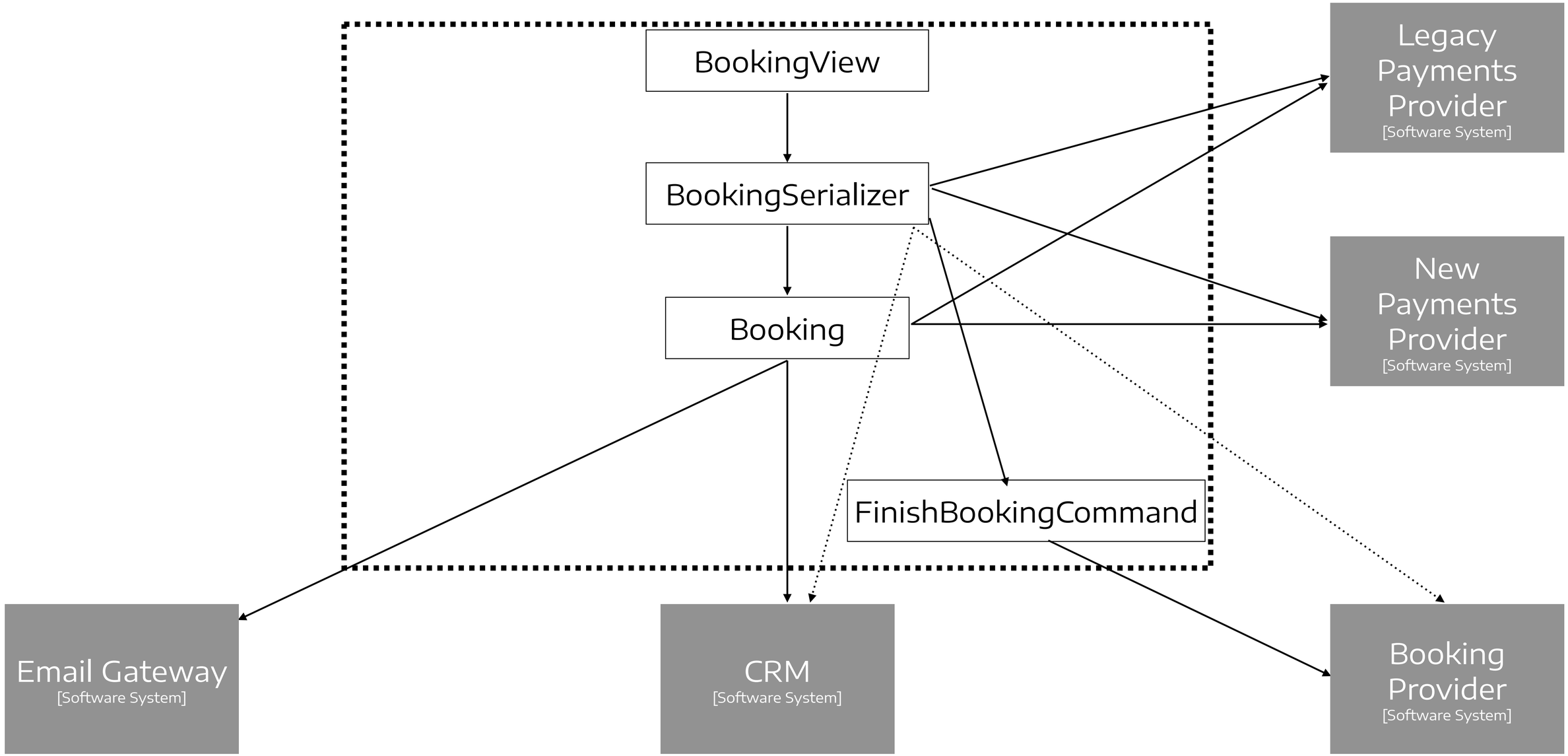
```python
def validate_payment_card_with_zero_auth(
        self, payment_card_token: str
    ) -> bool:
        if waffle.switch_is_active('use_new_payment'):
            try:
                madeup_payment.Charge.create( ... )
            except madeup_payment.PaymentFailed:
                return False

            return True
        else:
            client = old_payment.Client()
            response = client.authorize( ... )
            if response.status == old_payment.Status.SUCCESS:
                return True
            else:
                return False
```

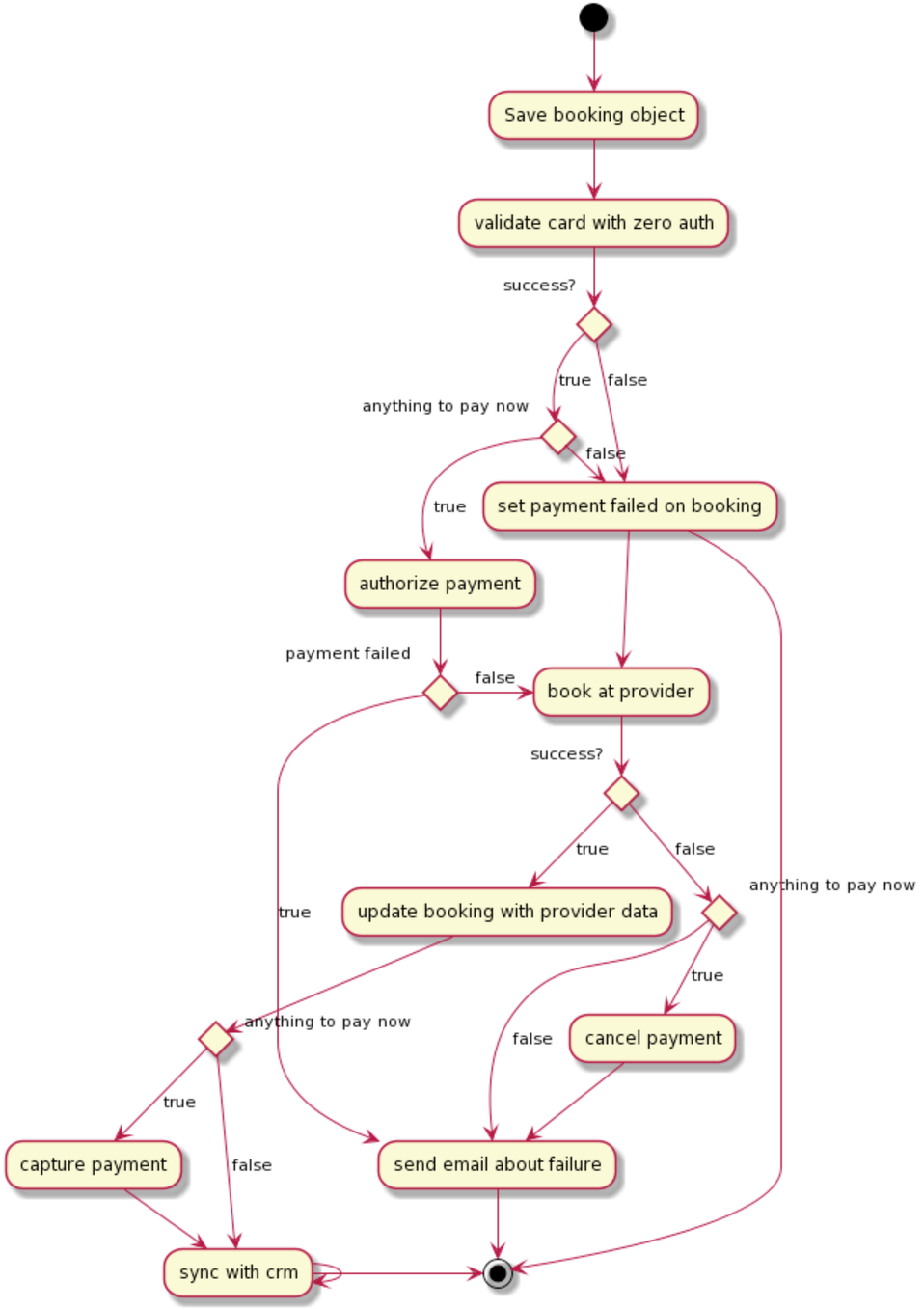
...how could this happen?

Accidental complexity

Made worse than it needs to be - refactor


Customer
[Person]
Makes bookings
Booking System
[Software System]
processes payments using
Legacy Payments Provider
[Software System]
processes payments using
New Payments Provider
[Software System]
sends e-mails using
records booking for reference
Fetches offers from and makes bookings using
Email Gateway
[Software System]
CRM
[Software System]
Booking Provider
[Software System]

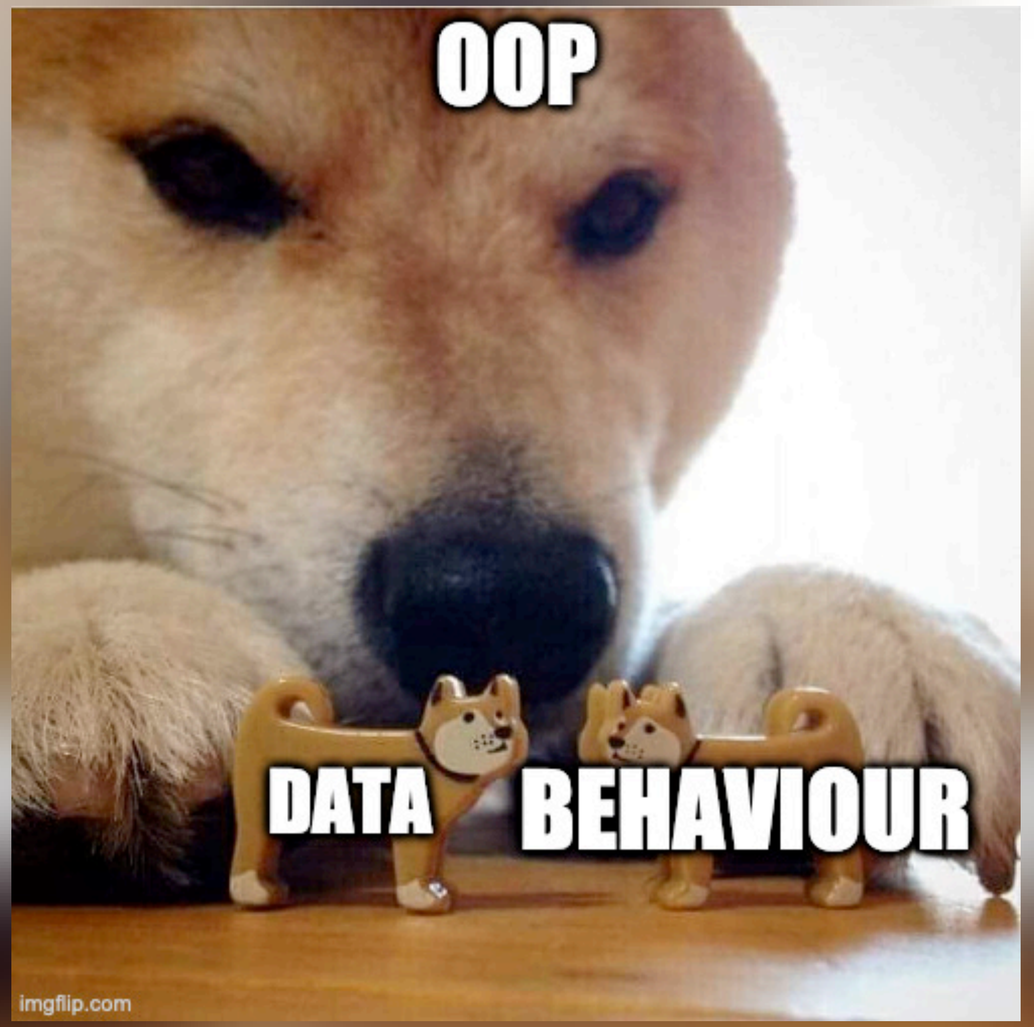

BookingView
BookingSerializer
Booking
FinishBookingCommand
Legacy Payments Provider
[Software System]
New Payments Provider
[Software System]
Email Gateway
[Software System]
CRM
[Software System]
Booking Provider
[Software System]


Save booking object
validate card with zero auth
success?
true
false
anything to pay now
false
true
set payment failed on booking
authorize payment
payment failed
false
book at provider
success?
true
false
anything to pay now
true
update booking with provider data
true
anything to pay now
false
cancel payment
true
capture payment
false
send email about failure
sync with crm

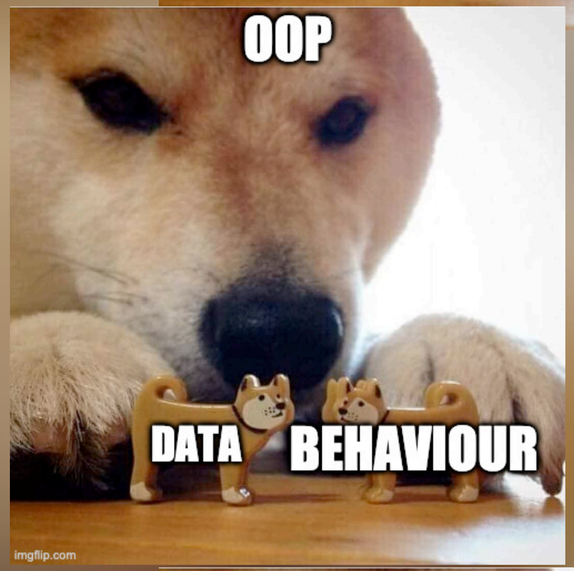
Essential complexity

No way to escape - one needs to manage it

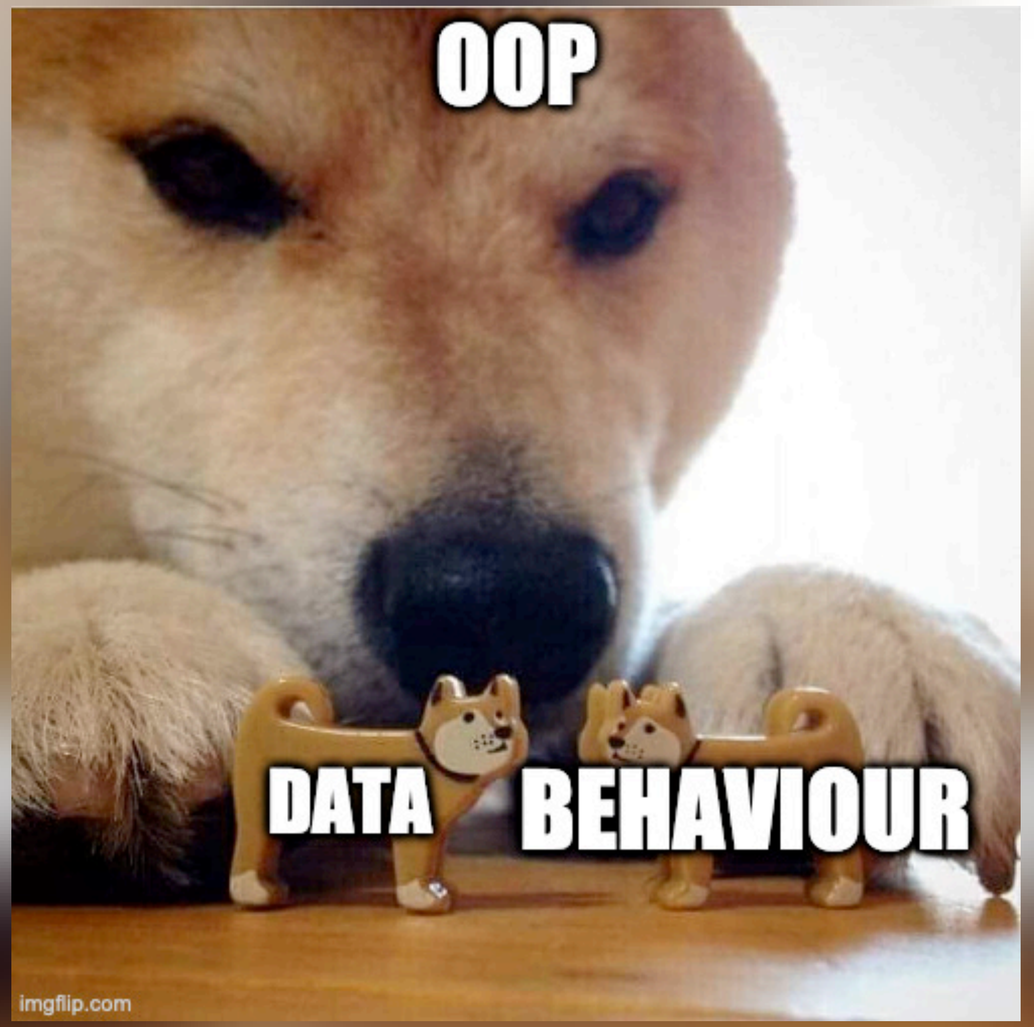
Object-Oriented Design to the rescue

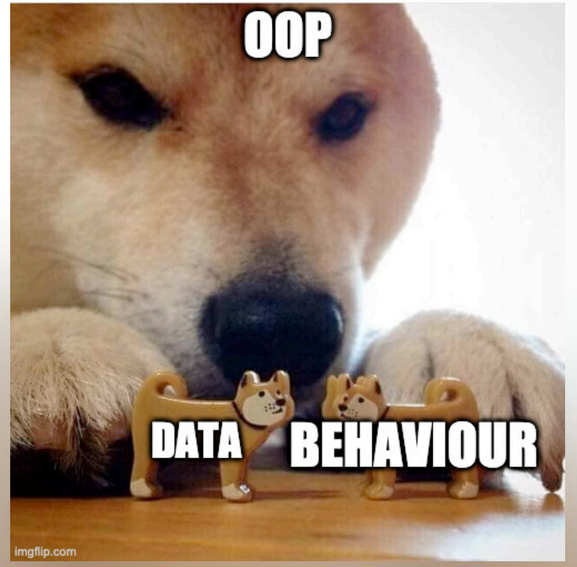

OOP
DATA
BEHAVIOUR
imgflip.com

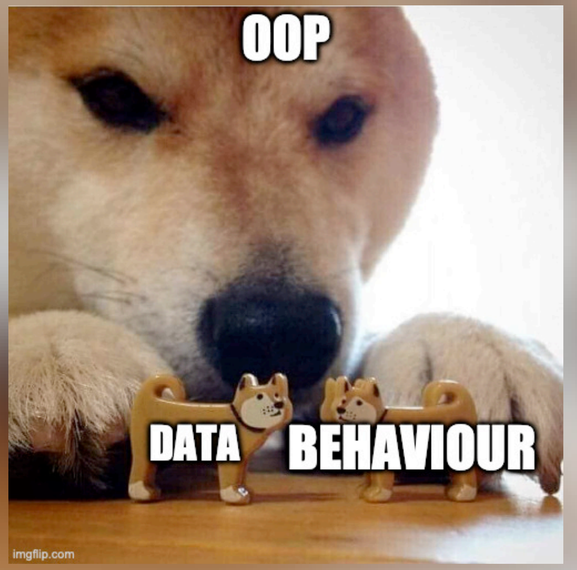

DATA
BEHAVIOUR

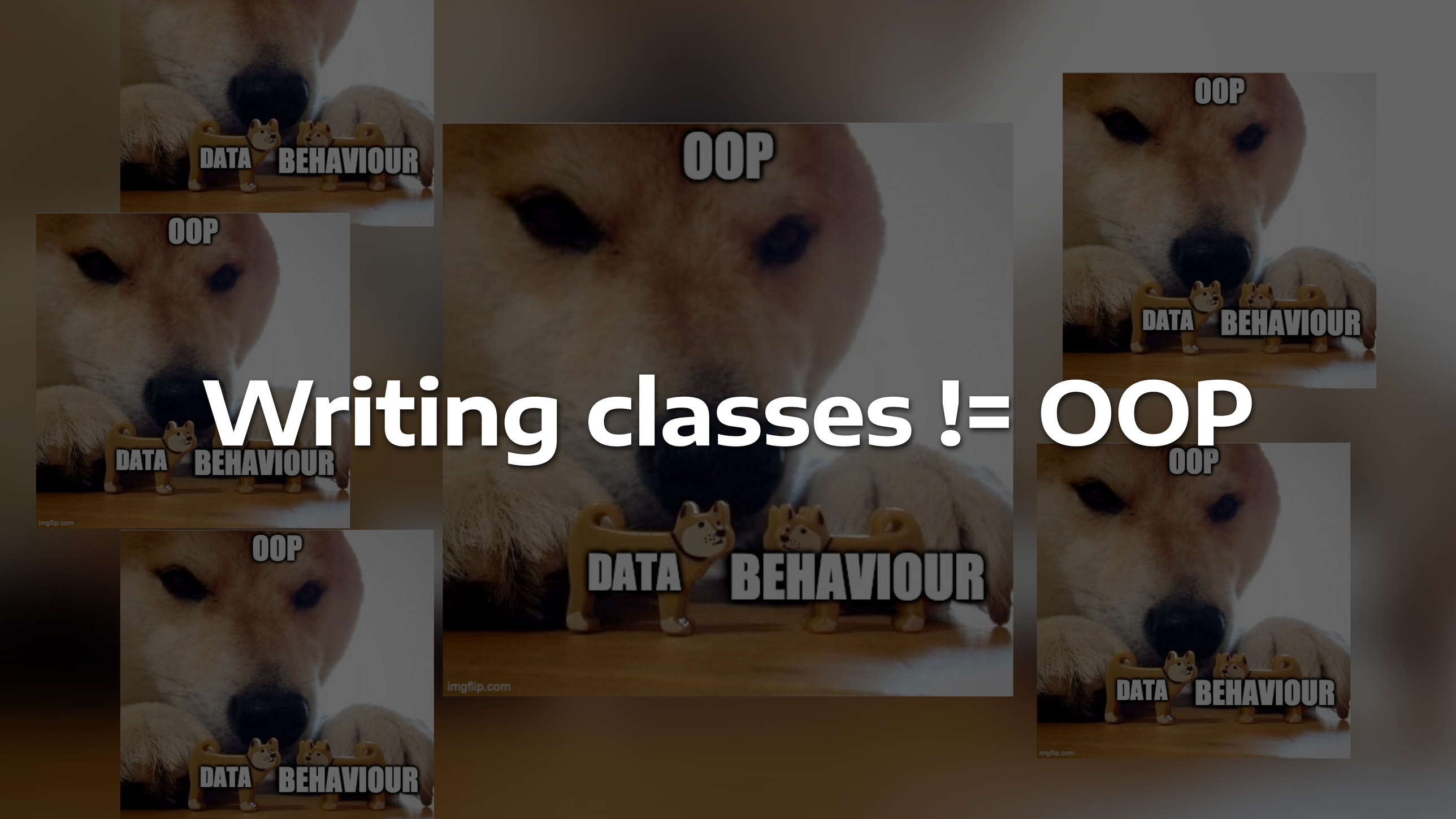

OOP
DATA
BEHAVIOUR
imgflip.com


OOP
DATA
BEHAVIOUR


OOP
DATA
BEHAVIOUR
imgflip.com


OOP
DATA
BEHAVIOUR
imgflip.com


OOP
DATA
BEHAVIOUR
imgflip.com



Writing classes != OOP

Object-Oriented design is like...

Cells creating an organism

Object-Oriented design is like...

Community members

Object-Oriented design is like...

Inheritance
Composition
Encapsulation
Abstraction
Polymorphism


SOLID
Tell, Don't Ask
WOW
GRASP
Law of Demeter
imgflip.com

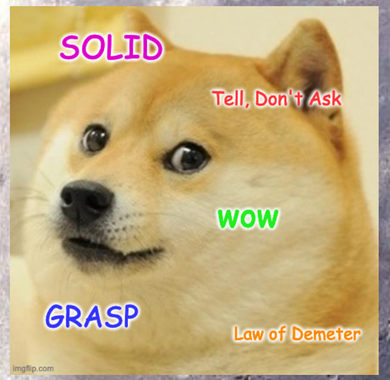
Inheritance

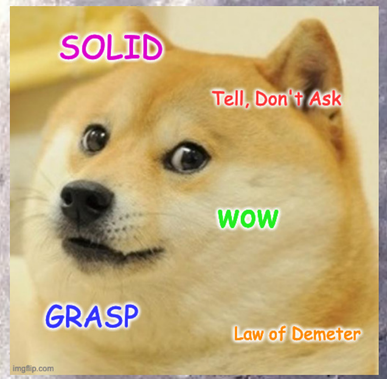
Composition

Encapsulation

Abstraction

Polymorphism





Inheritance

Composition

Encapsulation

Abstraction

Polymorphism

Responsibility-Driven Design

Roles

Responsibilities

Collaborations

Roles

Responsibilities

Collaborations

Roles

Responsibilities

Collaborations

Knowing

Doing

Deciding

Roles
Responsibilities
Collaborations

„Where to put that code?”

Role Stereotypes


SHREK
www.charlotte.com/justgo/movies/

Information Holder

Structurer

Service Provider

Coordinator

Controller

Interfacer

Information Holder

Structurer

Service Provider

Coordinator

Controller

Interfacer

Information Holder

Knows and provides information

```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) → Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)

    def fail_payment(self) → None:
        assert self.status is None
        self.status = self.Status.FAILED_PAYMENT
```

Information Holder

Knows and provides information

```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) → Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)

    def fail_payment(self) → None:
        assert self.status is None
        self.status = self.Status.FAILED_PAYMENT
```

Information Holder

Knows and provides information

```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) → Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)

    def fail_payment(self) → None:
        assert self.status is None
        self.status = self.Status.FAILED_PAYMENT
```

Information Holder

Knows and provides information

```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) → Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)

    def fail_payment(self) → None:
        assert self.status is None
        self.status = self.Status.FAILED_PAYMENT
```

Information Holder

Knows and provides information

```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) -> Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)

    def fail_payment(self) → None:
        assert self.status is None
        self.status = self.Status.FAILED_PAYMENT
```

Law of Demeter

Information Holder

Knows and provides information

```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) → Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)

    def fail_payment(self) → None:
        assert self.status is None
        self.status = self.Status.FAILED_PAYMENT
```

Information Holder

Knows and provides information

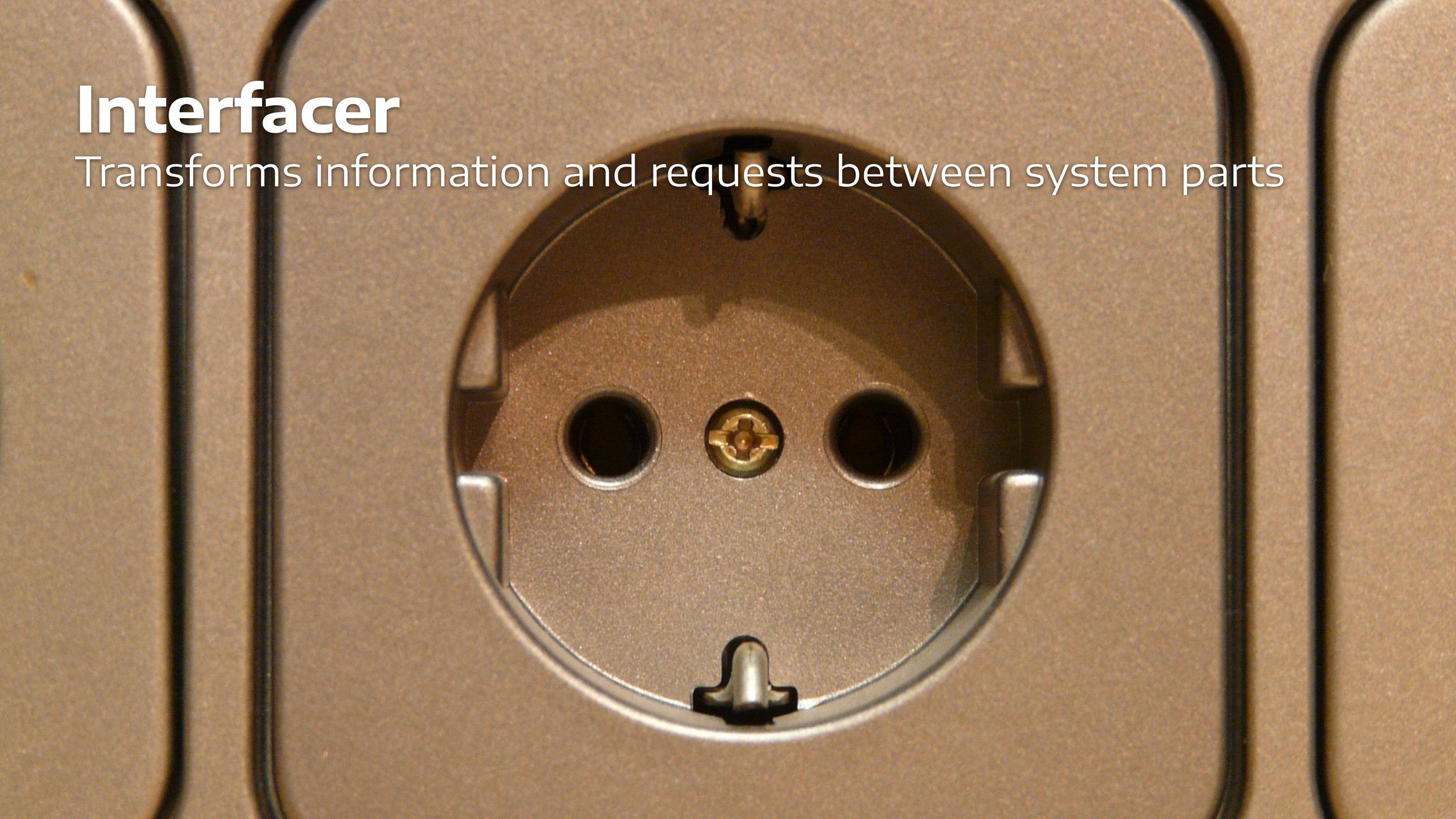
```
class Booking(models.Model):
    status = models.CharField( ... )

    @property
    def total(self) -> Money:
        all_prebookings = itertools.chain( ... )
        return sum(prebooking.total for prebooking in all_prebookings)
```

```
def fail_payment(self) -> None:
    assert self.status is None
    self.status = self.Status.FAILED_PAYMENT
```

Tell, Don't Ask

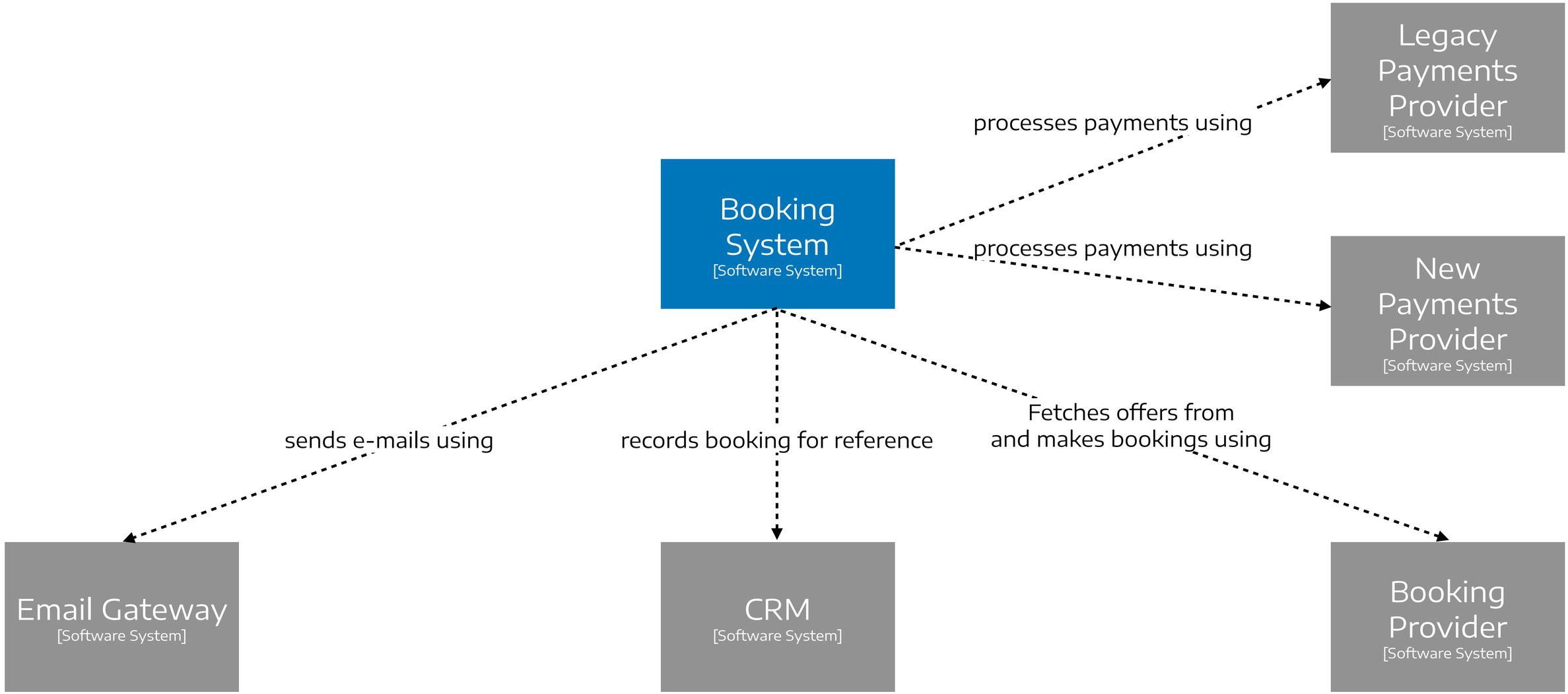
Interfacer

Transforms information and requests between system parts



Interfacer

Transforms information and requests between system parts



```python
def validate_payment_card_with_zero_auth(
        self, payment_card_token: str
    ) -> bool:
        if waffle.switch_is_active('use_new_payment'):
            try:
                madeup_payment.Charge.create( ... )
            except madeup_payment.PaymentFailed:
                return False

            return True
        else:
            client = old_payment.Client()
            response = client.authorize( ... )
            if response.status == old_payment.Status.SUCCESS:
                return True
            else:
                return False
```

Controller

Closely directs the action of other objects

Controller

Closely directs the action of other objects

```
class BookingSerializer(HyperlinkedModelSerializer):
    def create(self, validated_data) → Booking:
        ...

    @staticmethod
    def finish_booking(self, validated_data) → Booking:
        ...


class FinishBookingCommand:
    def finish_booking(self, booking: Booking) → None:
        ...
```

Disclaimer: Not trying to work against Django!

Information Holder

model

Structurer

Service Provider

Coordinator

signal handler

Controller

view

Interfacer


Marszałkowska
Poznańska
Poznańska

1. Introduce Interfacers

1. Introduce Interfacers
2. Consolidate Controller role

1. Introduce Interfacers
2. Consolidate Controller role
3. Strip side effects off Information Holders

Interfacer

Easy to use from Controller, uniform way to handle errors

```python
class Payments(abc.ABC):

    @abc.abstractmethod
    def perform_zero_auth(self, token: str) → None:
        pass


class AuthorizationFailed(Exception):
    pass
```

Interfacer

Not always need an abstract class!

```python
# crm.py
def sync_booking(booking: Booking) → None:
    ...
```

Interfacer

You just got a stable patching/mocking/stubbing point!

```
def sync_booking(booking: Booking) → None:
    ...


class Payments(abc.ABC):

    @abc.abstractmethod
    def perform_zero_auth(self, token: str) → None:
        pass
```

Controller

Whole flow in one place - merge FinishBooking with BookingSerializer

```
class BookingSerializer(ModelSerializer):
    def create(self, validated_data) → Booking:
        ...

        try:
            payments.perform_zero_auth(payment_card_token)
        except AuthorizationFailed:
            ...

        if booking.needs_pay_anything_now:  ...

        try:
            booking_api_response = madeup_client.book_at_once(references)
        except madeup_booking.MadeUpApiProviderError:
            ...
```

Controller

Whole flow in one place - merge FinishBooking with BookingSerializer

```
class BookingSerializer(ModelSerializer):
    def create(self, validated_data) → Booking:
        ...

        try:
            payments.perform_zero_auth(payment_card_token)
        except AuthorizationFailed:
            ...

        if booking.needs_pay_anything_now:  ...

        try:
            booking_api_response = madeup_client.book_at_once(references)
        except madeup_booking.MadeUpApiProviderError:
            ...
```

Controller

Whole flow in one place - merge FinishBooking with BookingSerializer

```
class BookingSerializer(ModelSerializer):
    def create(self, validated_data) → Booking:
        ...

        try:
            payments.perform_zero_auth(payment_card_token)
        except AuthorizationFailed:
            ...

        if booking.needs_pay_anything_now:  ...

        try:
            booking_api_response = madeup_client.book_at_once(references)
        except madeup_booking.MadeUpApiProviderError:
            ...
```

Controller

Whole flow in one place - merge FinishBooking with BookingSerializer

```
class BookingSerializer(ModelSerializer):
    def create(self, validated_data) → Booking:
        ...

        try:
            payments.perform_zero_auth(payment_card_token)
        except AuthorizationFailed:
            ...

        if booking.needs_pay_anything_now:  ...

        try:
            booking_api_response = madeup_client.book_at_once(references)
        except madeup_booking.MadeUpApiProviderError:
            ...
```

Controller

Whole flow in one place - merge FinishBooking with BookingSerializer

```
class BookingSerializer(ModelSerializer):
    def create(self, validated_data) → Booking:
        ...

        try:
            payments.perform_zero_auth(payment_card_token)
        except AuthorizationFailed:
            ...

        if booking.needs_pay_anything_now:  ...

        try:
            booking_api_response = madeup_client.book_at_once(references)
        except madeup_booking.MadeUpApiProviderError:
            ...
```

Controller

Introducing Service

```
class BookingsViewSet(CreateModelMixin, RetrieveModelMixin, GenericViewSet):
    serializer_class = BookingSerializer

    def perform_create(self, serializer):
        card_token = serializer.validated_data.pop('payment_card_token')
        booking: Booking = serializer.save()

        booking_client =  ...
        payments =  ...
        service = BookingService(payments, booking_client)

        service(booking, card_token)
```

Controller

Introducing Service

```
class BookingsViewSet(CreateModelMixin, RetrieveModelMixin, GenericViewSet):
    serializer_class = BookingSerializer

    def perform_create(self, serializer):
        card_token = serializer.validated_data.pop('payment_card_token')
        booking: Booking = serializer.save()

        booking_client = ...
        payments = ...
        service = BookingService(payments, booking_client)

        service(booking, card_token)
```

Controller

Introducing Service

```
class BookingsViewSet(CreateModelMixin, RetrieveModelMixin, GenericViewSet):
    serializer_class = BookingSerializer

    def perform_create(self, serializer):
        card_token = serializer.validated_data.pop('payment_card_token')
        booking: Booking = serializer.save()

        booking_client =  ...
        payments =  ...
        service = BookingService(payments, booking_client)

        service(booking, card_token)
```

Controller

Introducing Service

```
@dataclass
class BookingService:
    _payments: Payments
    _booking_client: BookingClient

    def __call__(self, booking: Booking, payment_card_token: str) → None:
        ...
```

Controller

Introducing Service

```python
@dataclass
class BookingService:
    _payments: Payments
    _booking_client: BookingClient

    def __call__(self, booking: Booking, payment_card_token: str) → None:
        ...
```

Composition

Also, encapsulation

Information Holder

Get rid of any side-effects-causing methods

```
-    def authorize_payment_at_creation(self, card_token: str):
-        ...
-
-    def send_email_about_failure(self, reason: str) → None:
-        ...
-
-    def cancel_payment(self):
-        ...
-
-    def capture_payment(self):
-        ...
-
-    def sync_with_crm(self):
-        ...
-
```

switching from procedural programing to object oriented programing be like

Object Design

Roles, Responsibilities, and Collaborations

Rebecca Wirfs-Brock and Alan McKean

Forewords by Ivar Jacobson and John Vlissides

How Designs Differ

http://www.wirfs-brock.com/PDFs/How%20Designs%20Differ.pdf

Heuristics

not commandments

Design is more like art

There are many good possible solutions

No long refactoring journeys

Do it daily

Sebastian Buczyński

breadcrumbscollector.tech

Origin of used resources

https://pixabay.com/photos/worker-grinder-factory-workplace-5736096/
https://pl.wikipedia.org/wiki/Ulica_Piotrkowska_w_%C5%81odzi#/media/Plik:Ulica_Piotrkowska_in_Lodz.JPG
https://pixabay.com/photos/step-path-shoes-sole-direction-780896/
https://pixabay.com/photos/chain-link-fence-metal-690503/
https://pixabay.com/photos/disposal-dump-garbage-junk-1846033/
https://photofunia.com/pl/effects/retro-wave + Django logo
https://pixabay.com/photos/compass-hand-holding-outdoors-1850673/
https://imgflip.com/meme/172118219/Shiba-Making-Toys-Kiss
https://pixabay.com/photos/virus-pathogen-antibody-antibodies-5741636/
https://pixabay.com/photos/commuters-busy-train-station-london-692137/
https://pixabay.com/photos/actor-show-stage-sideshow-theater-779472/
https://imgflip.com/meme/Doge
https://pixabay.com/photos/moon-sky-luna-craters-lunar-1527501/
Shrek (Movie)
Lord of the Rings (Movie)
https://pixabay.com/photos/socket-plug-current-electricity-5504/
https://pixabay.com/photos/conductor-andrea-vitello-concert-5157150/
https://pixabay.com/photos/map-navigation-hands-travel-route-455769/