xxx Python Developer recruitment task

Background

We want to develop a ticket-selling platform. We are hoping to get popular pretty quickly, so prepare for *high traffic*!
The front-end part of the application is not yet ready, so feel free to design the API however you want.

High-level features

1. Get info about an event
2. Get info about available tickets
3. Reserve ticket
4. Pay for ticket
5. Get info about reservation
6. Get reservation statistics

Feature requirements

Get info about an event

1. Event has a name
2. Event has a date and a time
3. Event can have multiple types of tickets (eg. regular, premium, VIP)

Get info about available tickets

- We should be able to receive information about which tickets are still available for sale and in which quantity.

Reserve ticket

- Reservation is valid for 15 minutes, after that it is released.

Pay for ticket

- For the sake of simplicity we operate only on the EUR currency. Feel free to use the provided adapter.

Get info about reservation

- Return information about the state of the reservation and its data

Get reservation statistics

- Get a summary amount of reserved tickets for each event
- Get a summary amount of reserved tickets of a specified type (eg. a total number of reserved VIP tickets)
- Feel free to add any other interesting statistics

Additional info

1. We really don't want to push you in any certain direction, but if you've made certain assumptions how it should work with FE then please let us know. TL;DR - you can briefly describe how the whole app would interact.
2. Please use a relational (SQL) database such as Postgres, MySQL or SQLite.
3. Goes without saying, but tests would be highly appreciated.

```python
from collections import namedtuple


class CardError(Exception):
    pass


class PaymentError(Exception):
    pass


class CurrencyError(Exception):
    pass


PaymentResult = namedtuple('PaymentResult', ('amount', 'currency'))


class PaymentGateway:
    supported_currencies = ('EUR',)

    def charge(self, amount, token, currency='EUR'):
        if token == 'card_error':
            raise CardError("Your card has been declined")
        elif token == 'payment_error':
            raise PaymentError("Something went wrong with your transaction")
        elif currency not in self.supported_currencies:
            raise CurrencyError(f"Currency {currency} not supported")
        else:
            return PaymentResult(amount, currency)
```