

Aineopintojen harjoitustyö:

Tietokantasovellus

Pizzapalvelu

Atte Keltanen

[Introduction](#)

[Use cases](#)

[Data content](#)

[Database diagram](#)

[Installation Instructions](#)

[Overview of the Project](#)

[Components](#)

[My Experiences](#)

Introduction

The goal of this project is to offer a pizza ordering service which will be secure, convenient, and easy for the customer to use. There will be support for kebabs and drinks as well.

The project will be done in English to target a wider audience.

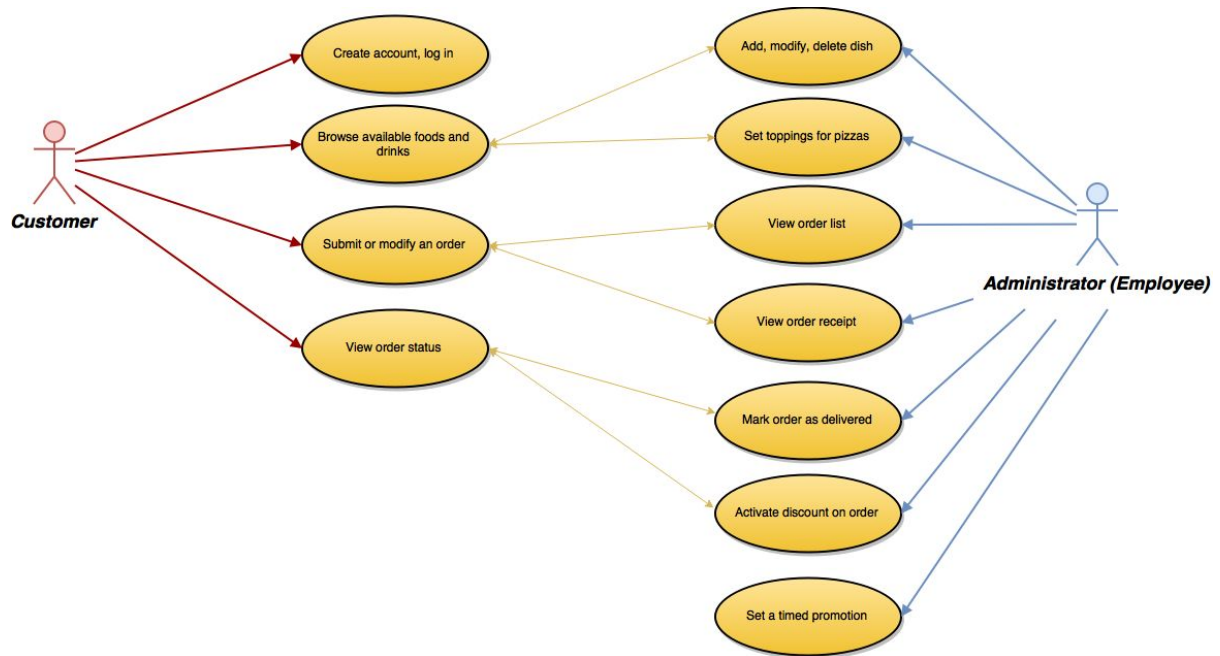
The back end will run on Python with Flask. The front end will very likely use React. PostgreSQL has been selected as the database of choice.

Heroku is supported out of the box and will be used to host the development version.

Use cases

User types:

- Customer: A customer who wants to order food.
- Administrator (employee): An administrator who will fulfill orders. In this case an employee.



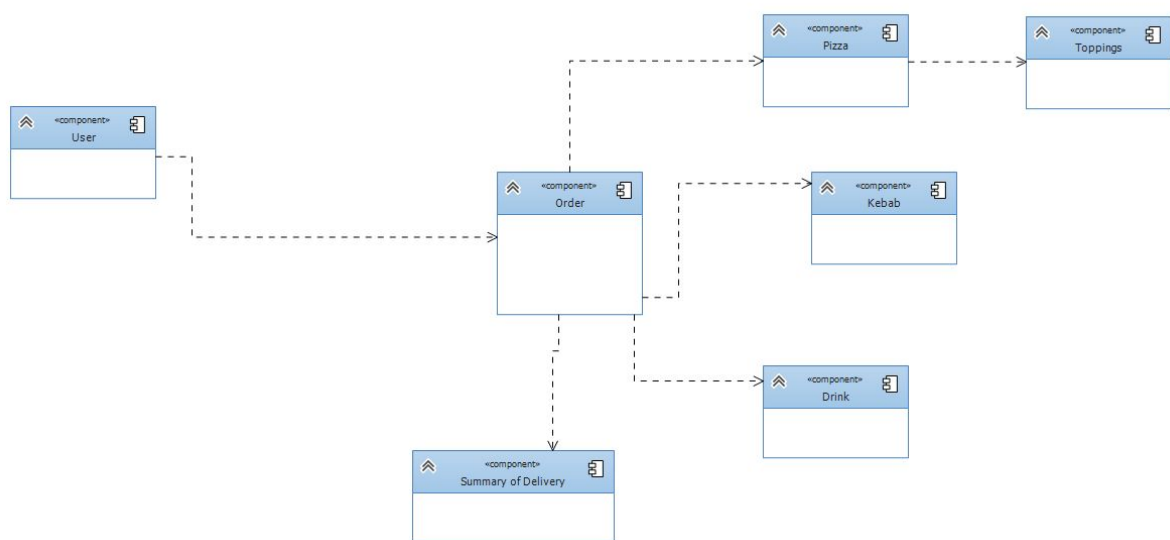
Customer:

- Account creation and logging in: The customer can create an account and log in with it.
- Browsing available pizzas, kebabs, and drinks: The customer can view the available offerings.
- Submitting and modifying an order: The customer can create an order with selected products and modify it if it hasn't been fulfilled yet.
- Viewing order status: The customer can see the status of their order.

Administrator:

- Adding, modifying, and deleting a dish: The administrator can add stuff for the user to order.
- Adding and setting toppings for pizzas: The administrator can tweak their pizzas.
- Viewing a list of active orders: The administrator can view what orders need to be fulfilled.
- Viewing an order receipt: The administrator can view the details of an order.
- Marking orders as delivered: The administrator can confirm orders as fulfilled.
- Activating discounts on orders: The administrator can add a discount on orders that were delivered late.
- Setting a timed promotion: The administrator can set a lunch offer on selected dishes.

Data content



Component: Pizza

Attribute	Value type	Description
Name	String (maximum of 512 characters)	Name of pizza 'bolognese'
Price	Money	Price of pizza
Image	String (maximum of 512 characters)	Image of pizza

Component: Topping

Attribute	Value type	Description
Name	String (maximum of 512 characters)	Name of topping
Price	Money	Price of topping

Component: Kebab

Attribute	Value type	Description
Name	String (maximum of 512 characters)	Name of kebab
Price	Money	Price of kebab
Image	String (maximum of 512 characters)	Image of kebab

Component: User

Attribute	Value type	Description
Username	String (maximum of 512 characters)	Username the user uses to log in
Password	String (maximum of 512 characters)	Password the user uses to log in
Registration date	Date with time	Timestamp when the user account has been created
Admin status	Boolean	Designates whether the user is considered an admin or not

Component: Drink

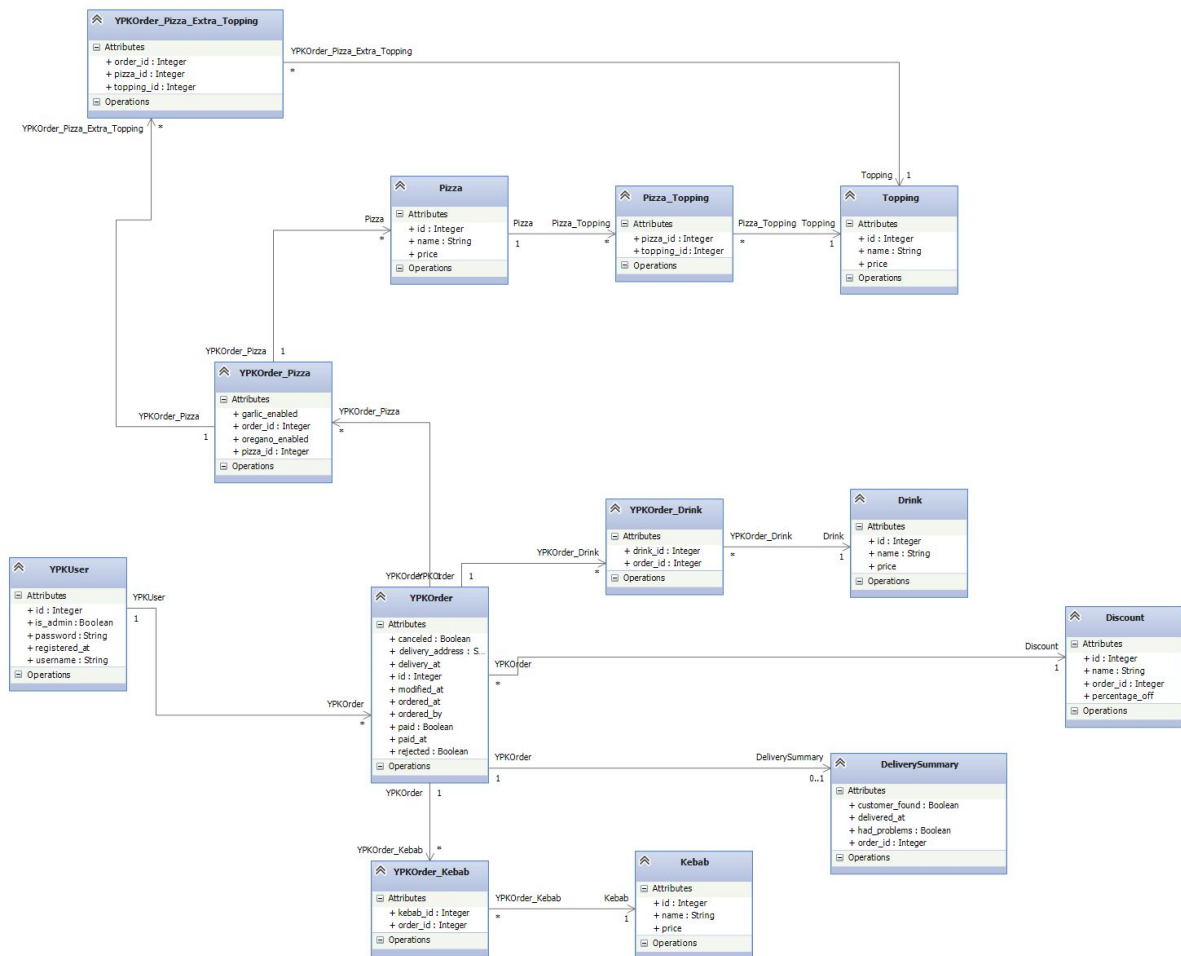
Attribute	Value type	Description
Name	String (maximum of 512 characters)	Name of drink
Price	Money	Price of drink
Image	String (maximum of 512 characters)	Image of drink

Component: Order

Attribute	Value type	Description
-----------	------------	-------------

Ordered at	Date with time	Timestamp of when the order has been created
Modified at	Date with time	Timestamp of when the order has been modified
Delivery address	String (maximum of 512 characters)	Address to deliver the order to
Delivery at	Date with time	Agreed delivery time
Canceled	Boolean	Set to true if the user has canceled the order
Rejected	Boolean	Set to true if the seller has rejected the order due to some reasons

Database diagram



All prices are in 'money' value type.

Modied_ats, ordered_ats, etc are timestamps with timezone.

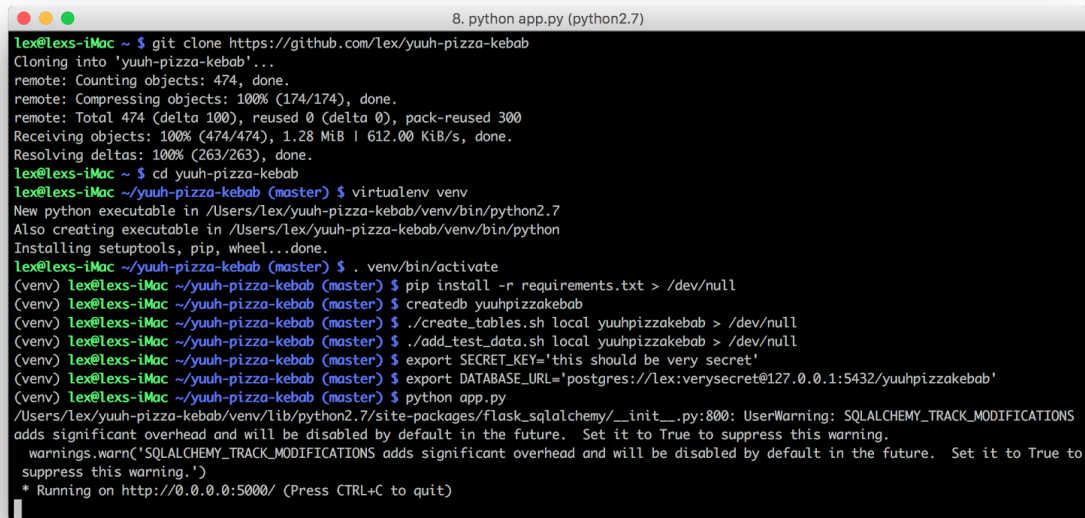
Ordered_by is a foreign key to YPKUser.

Percentage_off is an integer.

There still may be some important things missing, but for now the database looks like this.

Installation Instructions

Clone the repository, create a database, create a virtual environment, install the dependencies, run the database setup scripts and run the application:

A terminal window titled '8. python app.py (python2.7)' showing the following commands and output:

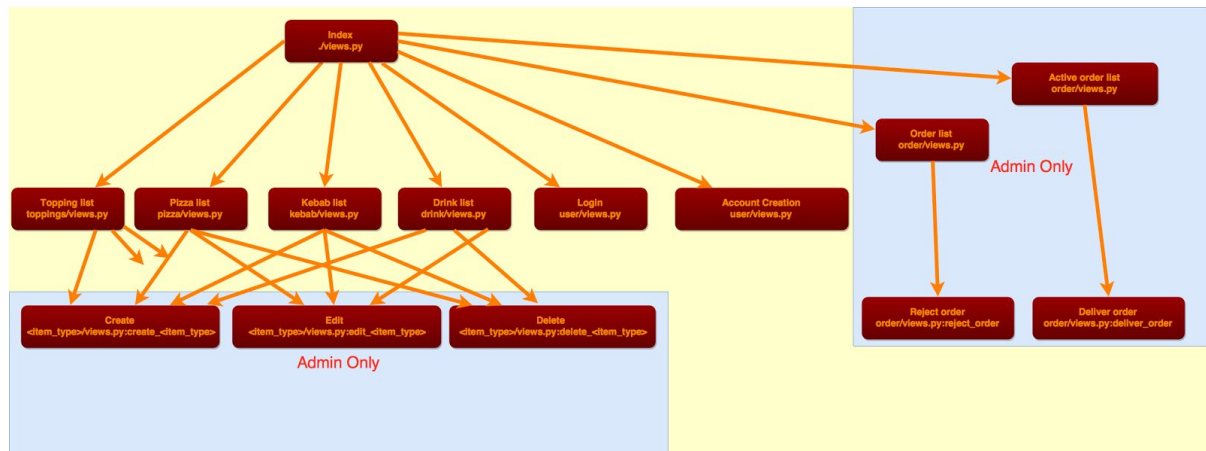
```
lex@lexs-iMac ~ $ git clone https://github.com/lex/yuuh-pizza-kebab
Cloning into 'yuuh-pizza-kebab'...
remote: Counting objects: 474, done.
remote: Compressing objects: 100% (174/174), done.
remote: Total 474 (delta 100), reused 0 (delta 0), pack-reused 300
Receiving objects: 100% (474/474), 1.28 MiB | 612.00 KiB/s, done.
Resolving deltas: 100% (263/263), done.
lex@lexs-iMac ~ $ cd yuuh-pizza-kebab
lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ virtualenv venv
New python executable in /Users/lex/yuuh-pizza-kebab/venv/bin/python2.7
Also creating executable in /Users/lex/yuuh-pizza-kebab/venv/bin/python
Installing setuptools, pip, wheel...done.
lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ . venv/bin/activate
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ pip install -r requirements.txt > /dev/null
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ createdb yuuhpizzakebab
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ ./create_tables.sh local yuuhpizzakebab > /dev/null
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ ./add_test_data.sh local yuuhpizzakebab > /dev/null
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ export SECRET_KEY='this should be very secret'
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ export DATABASE_URL='postgres://lex:verysecret@127.0.0.1:5432/yuuhpizzakebab'
(venv) lex@lexs-iMac ~/yuuh-pizza-kebab (master) $ python app.py
/Users/lex/yuuh-pizza-kebab/venv/lib/python2.7/site-packages/flask_sqlalchemy/__init__.py:800: UserWarning: SQLALCHEMY_TRACK_MODIFICATIONS
adds significant overhead and will be disabled by default in the future. Set it to True to suppress this warning.
warnings.warn('SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True to
suppress this warning.')
```

Overview of the Project

The project is basically structured like a Django project. There are modules (directories) for every model. Every directory has at least two files: `models.py` and `views.py`. `views.py` acts like a controller like in Django.

Templates are in the `templates` directory, and model specific templates are inside directories named after them.

Components



My Experiences

The project itself was pretty easy since I've used all of these tools before this course.

The hardest part was creating time to make progress with the project. That means I had to spend most of my weekends on it. The hardest part code wise was creating the ordering process without any additional JavaScript. It stores everything in the session instead, which means there has to be a lot of different endpoints to handle all the stuff needed.

I learned some tricks with the Jinja2 templates and how some things in Bootstrap work.

The most interesting thing was when I realized I was practically making a new Django with the project structure I started to pursue in the very beginning. I don't know if it was a coincidence or not.