Кратко описать предметную область, концепцию и задачи (функционал) приложения текстом.

Проект «доставка еды»

Сущности процесса:

* Ресторан (Restaurant);
* Заказ (Order);
* Клиент (Client),
* Доставка (Delivery).

Накладываем ограничения и формализируем связь между сущностями:

* Клиент делает заказ;
* Ресторан заказ выполняет;
* Доставка заказ доставляет;
* У Ресторана может быть сколько угодно Заказов;
* У Клиента может быть сколько угодно Заказов.
* У Доставщика только готовые Заказы.

Таким образом, Ресторан, Клиент и Доставщик связаны через сущность Заказ.

Функционал приложения:

* Клиент может создавать Заказы, Ресторан и Доставщик заказ обрабатывает. Клиенты могут просматривать свои заказы и оценивать их.

Реализация:  
RESTful API приложение с помощью Python и библиотеки Flask, на базе паттерна MVC. База данных Postgresql

|  |  |  |
| --- | --- | --- |
| **Метод HTTP** | **URI** | **Действие** |
| POST | http://[hostname]/todo/api/v1.0/client | Создать заказ |
| GET | http://[hostname]/todo/api/v1.0/client | Посмотреть заказы клиента |
| PUT | http://[hostname]/todo/api/v1.0/client/[order\_id] | Оценить заказ |
| GET | http://[hostname]/todo/api/v1.0/restaurant/orders | Посмотреть заказы в ресторане |
| PUT | http://[hostname]/todo/api/v1.0/restaurant/orders/[order\_id] | Изменить заказ (взять в работу, выполнить заказ) |
| GET | http://[hostname]/todo/api/v1.0/delivery/orders | Посмотреть заказы доставки |
| PUT | http://[hostname]/todo/api/v1.0/delivery/orders/[order\_id] | Изменить заказ (взять в работу, выполнить заказ) |

Решением для реализации доступа к данным будет паттерн **Data Mapper**.

Каждый объект имеет свою зону ответственности, тем самым следую принципам SOLID и сохраняя каждый объект простым и по существу.

Бизнес-логика и сохранение данных связаны слабо, и если вы хотите сохранять данные в XML-файл или какой-нибудь другой формат, вы можете просто написать новый Mapper, не притрагиваясь к доменному объекту.