# Проектирование

Целью проектирования является определение того, как будет реализована функциональность логической модели, построенной на этапе анализа. Анализ предметной области завершён, создана аналитическая модель.

## Проектные классы

Проектные классы – это классы, описание которых настолько полно, что они могут быть реализованы [умл2]. Описание формируется путем уточнения классов анализа, которое включает в себя добавлений деталей реализации. На рисунке 1 представлена часть диаграммы проектных классов, реализуемых в системе. Эта часть диаграммы отображает группу классов, относящихся к «Заведениям».

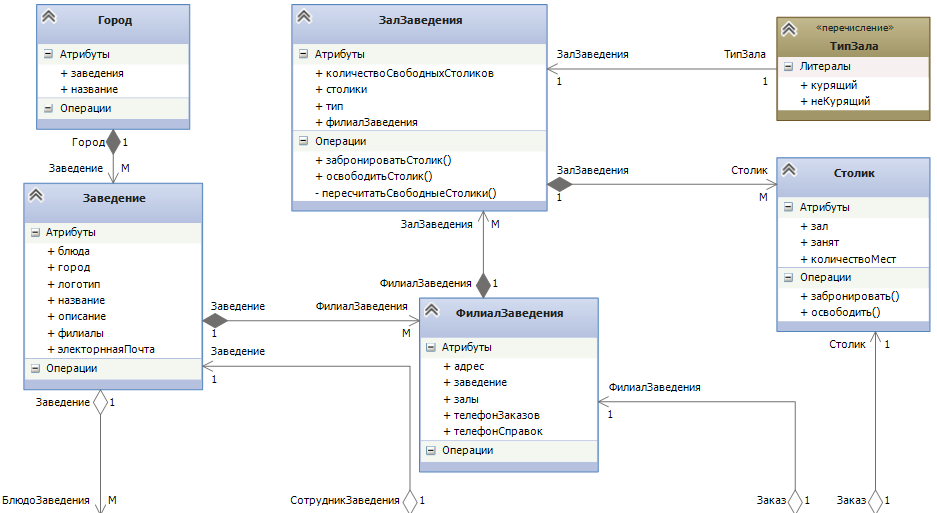


Рисунок 1 – Диаграмма проектных классов

Если сравнивать эту часть диаграммы с аналогичной частью диаграммы классов анализа, то очевидно, что изменения минимальны, а именно, объекты класса «Город» приобрели новое свойство – «Заведения», в котором содержится список заведений города. Еще одним изменением является уточнение отношений «Заведение» - «Блюдо Заведения» (агрегация, один ко многим), «Заведение» - «Сотрудник Заведения» (агрегация, один к одному), «Филиал» - «Заказ» (агрегация, один ко многим), «Заказ» - «Столик» (агрегация, один к одному).

На рисунке 2 представлена часть диаграммы, которая отображает группу классов, относящихся к «Блюдам».

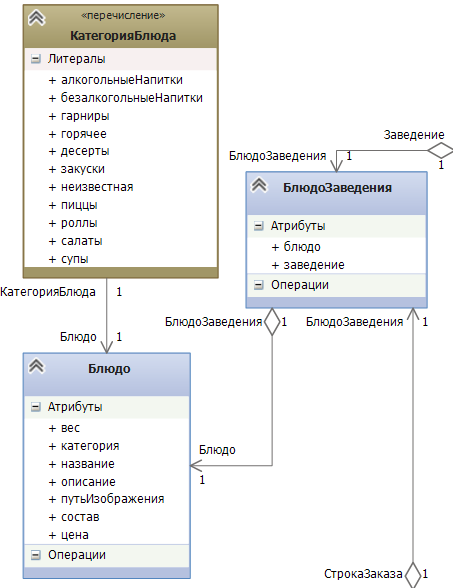


Рисунок 2 – Диаграмма проектных классов

Эта часть также не претерпела сильных изменений, помимо уже указанных были изменены отношения «Блюдо Заведения» - «Блюдо» (агрегация, один к одному), «Строка заказа» - «Блюдо Заведения» (агрегация, один к одному). Здесь необходимо сказать о том, что классы «Корзина Заказов» и «Строка Корзины Заказов» были заменены на «Строка Заказа». Это связано с тем, что было принято решение хранить «Корзину» в сессии.

На рисунке 3 представлена часть диаграммы, которая отображает группу классов, относящихся к «Заказам».

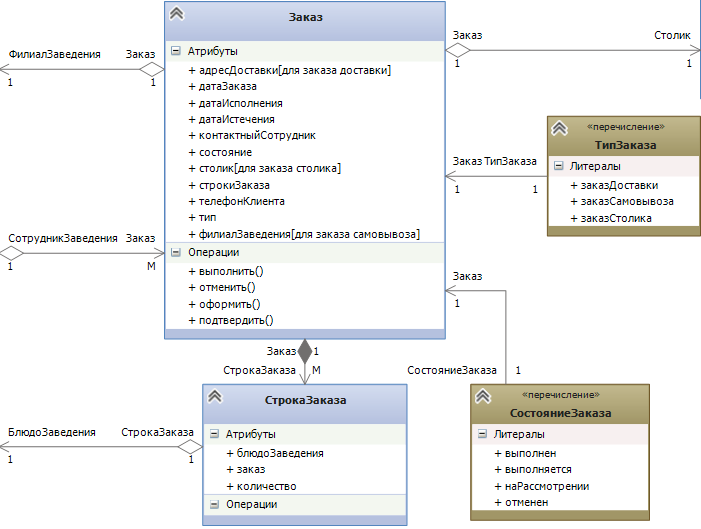


Рисунок 3 – Диаграмма проектных классов

Изменения в этой части более существенны, так как, как уже указывалось, «Корзина Заказов» была вынесена в сессию. Поэтому добавился класс «Строка Заказов», а также добавились и изменились отношения: «Заказ» - «Строка Заказов» (композиция, один ко многим), «Сотрудник Заведения – «Заказ» (композиция, один ко многим).

На рисунке 4 представлена последняя часть диаграммы, которая отображает группу классов, относящихся к «Управлению Заведений».

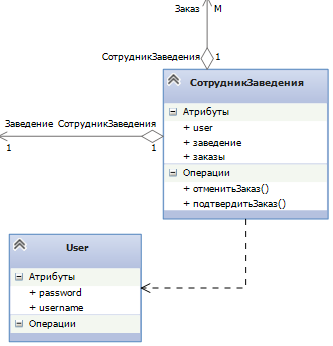


Рисунок 4 – Диаграмма проектных классов

Здесь было решено добавить класс «User». Это класс Django, который необходим для авторизации, от него зависят все пользователи системы.

## Диаграмма пакетов системы

Пакет (package) – это инструмент группирования, который позволяет взять любую конструкцию UML и объединить ее элементы в единицы высокого уровня. При этом каждый элемент (класс) может принадлежать только одному пакету. Пакеты же могут входить в состав других пакетов [http://www.planerka.info/item/Diagrammy-paketov].

На рисунке 5 представлена диаграмма пакетов. Она состоит из пакетов: «Заведения», «Блюда», «Заказы», «Управление Заведений».

Пакет «Заведения» содержит все классы, содержащие информацию о заведении (местоположение и планировка) – для ориентации в заведениях, бронирования столиков, самовывоза. Этот пакет зависит от пакета «Блюда». От этого пакета зависят пакеты «Заказы» и «Управление Заведений».

Пакет «Блюда» содержит классы, содержащие информацию о блюдах, а также класс, связывающий блюда с заведением. От этого пакета зависят пакеты «Заведения» и «Заказы».

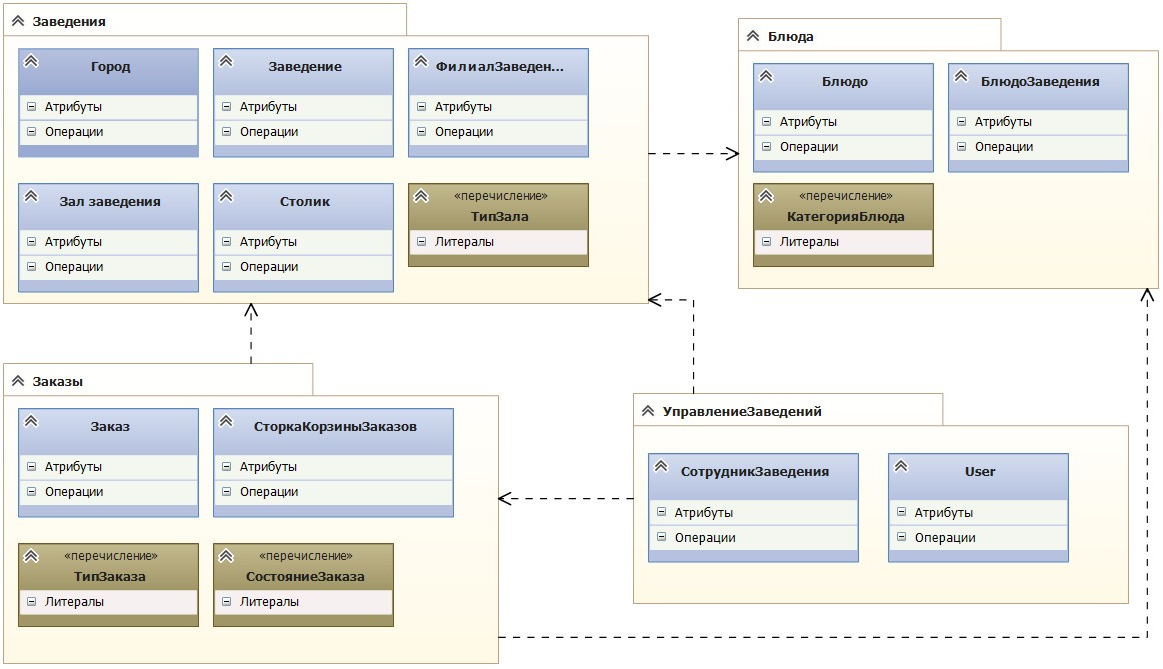


Рисунок 5 – Диаграмма пакетов

Пакет «Заказы» содержит классы для создания и хранения заказов. Этот пакет зависит от пакетов «Заведения» и «Блюда».

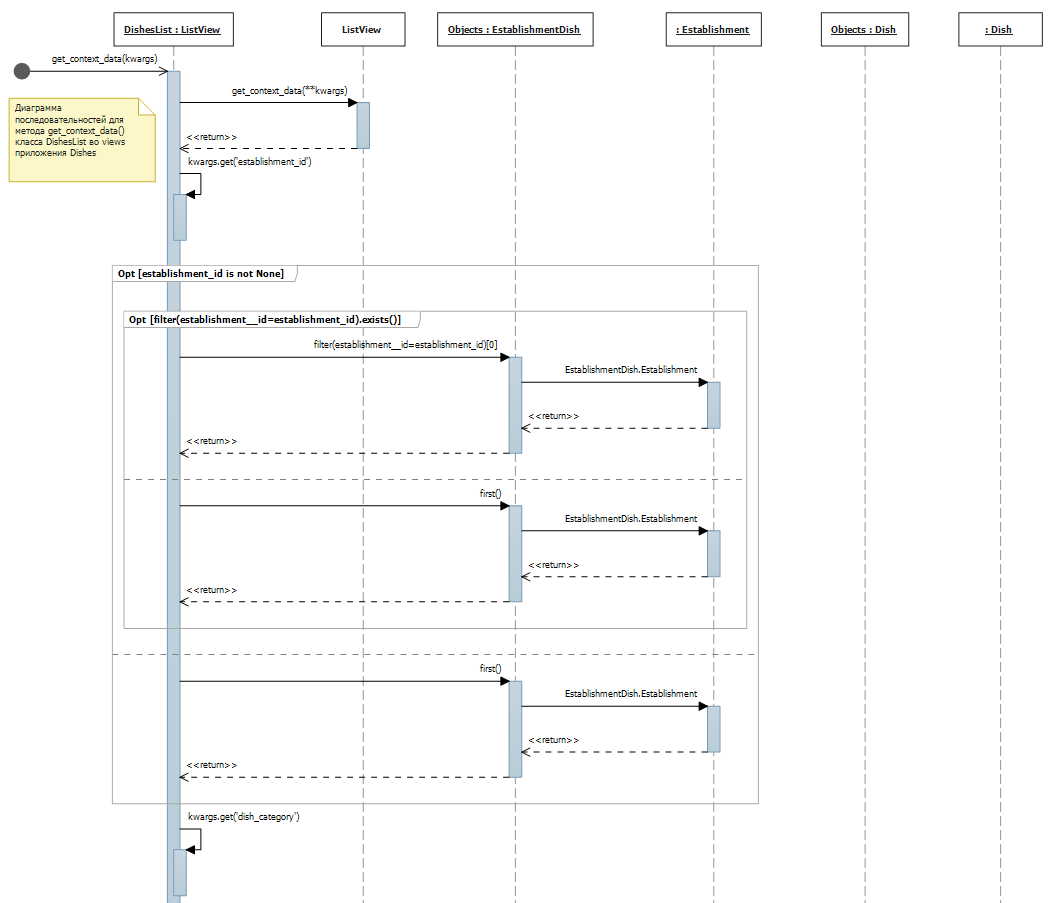
Пакет «Управление заведений» содержит класс, объект которого – аккаунт сотрудника, который может просматривать, подтверждать или отменять заказы. Этот пакет зависит от пакета «Заведения».

Пакет SQL database является базой данных, используемой для хранения информации, необходимой для работы системы.

## Диаграммы последовательностей для операций проектных классов

Диаграмма последовательности (англ. sequence diagram) — диаграмма, на которой показано взаимодействие объектов (обмен между ними сигналами и сообщениями), упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления [https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0\_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8]. Неотъемлемой частью объекта на диаграмме последовательности является линия жизни объекта. Линия жизни показывает время, в течение которого объект существует в Системе. Периоды активности объекта в момент взаимодействия показываются с помощью фокуса управления. Временная шкала на диаграмме направлена сверху вниз [http://it-gost.ru/articles/view\_articles/94].

В ходе выполнения курсового проекта были спроектированы диаграммы последовательностей для нескольких функций. На рисунке 6 изображена диаграмма последовательности для метода get\_context\_data(self, \*\*kwargs) класса DishesList во views приложения Dishes.



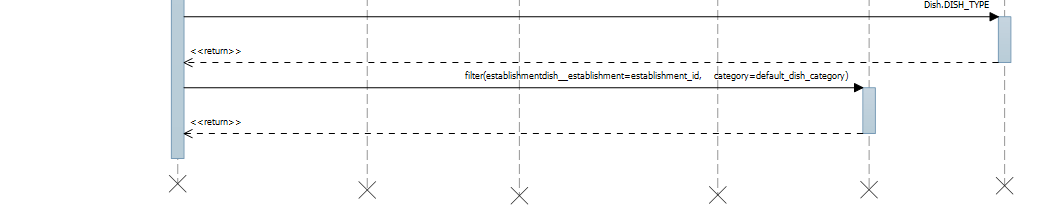


Рисунок 6 – Диаграмма последовательности get\_context\_data(self, \*\*kwargs) в DishesList

При вызове get\_context\_data(self, \*\*kwargs) происходит создание нового context, которому передается результат вызова super().get\_context\_data(\*\*kwargs). После этого задается параметр establishment\_id=self.kwargs.get('establishment\_id'). Далее осуществляется двойная проверка: первая – что в establishment\_id есть значение, вторая – фильтрует заведения по id и возвращает True, если заведение найдено. Если обе проверки пройдены успешно, то переменной current\_establishment присваивается найденный объект-заведение, иначе присваивается первое заведение из списка. Далее задается переменная dish\_category=self.kwargs.get('dish\_category'), после чего происходит получение списка Dish.DISH\_TYPE и осуществляется фильтрация по establishment\_id и default\_dish\_category, с целью получения списка блюд указанной категории в определенном заведении.

На рисунке 7 изображена диаграмма последовательности для метода get\_context\_data(self, \*\*kwargs) класса EstablishmentList во views приложения Establishments.

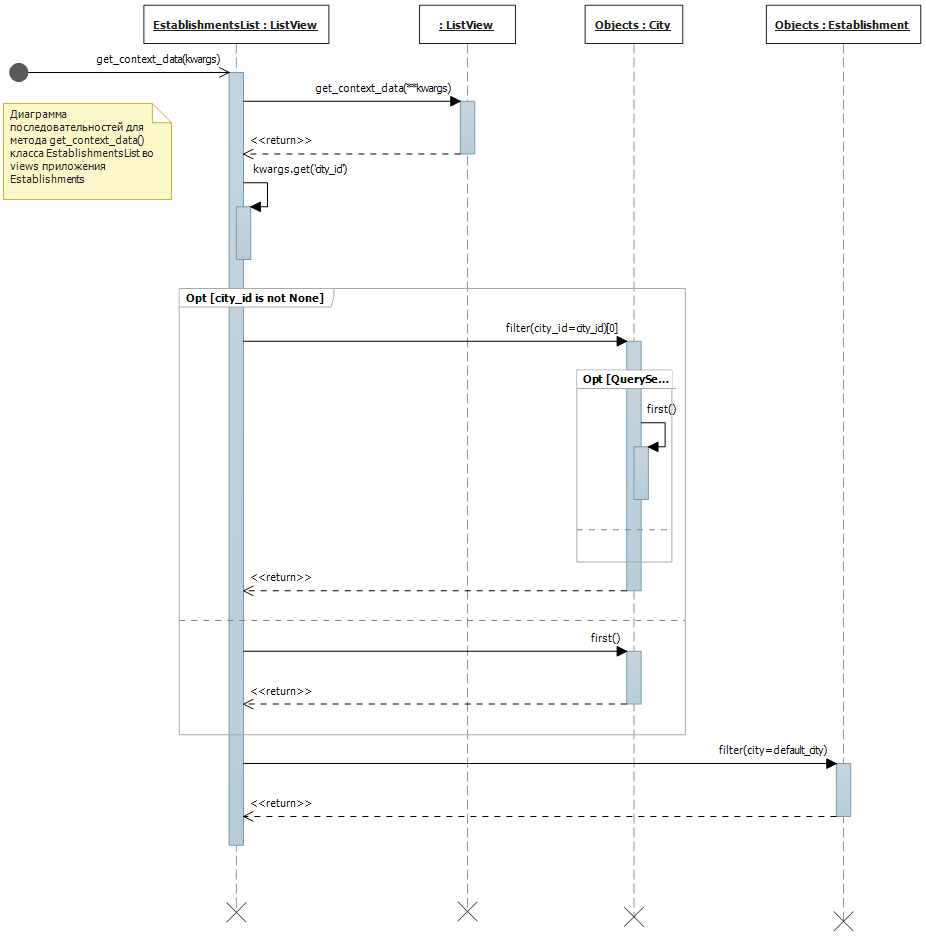


Рисунок 7 – Диаграмма последовательности get\_context\_data(self, \*\*kwargs) в EstablishmentList

При вызове get\_context\_data(self, \*\*kwargs) происходит создание нового context, которому передается результат вызова super().get\_context\_data(\*\*kwargs). После этого задается параметр city\_id=self.kwargs.get(city\_id'). Далее осуществляется проверка того, есть ли в city\_id значение. Если значение есть, то переменной default\_city присваивается найденный по фильтру или первый объект-город, иначе присваивается первый город из списка. Затем по фильтру city=default\_city возвращается список заведений города.

На рисунке 8 изображена диаграмма последовательности для метода increment\_dish(request) во views приложения Orders.

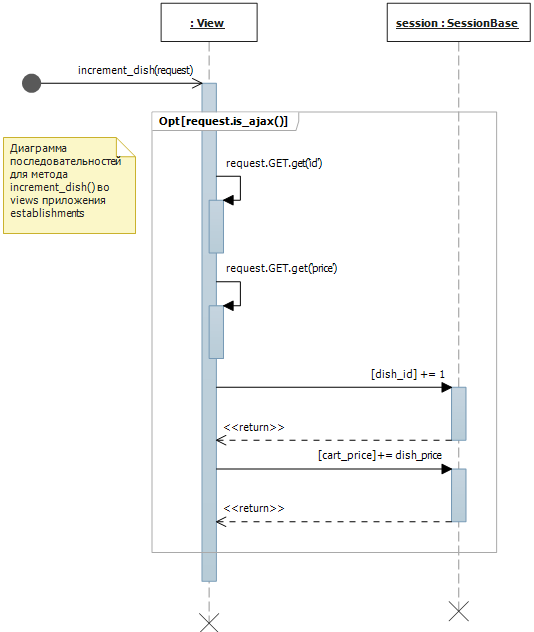
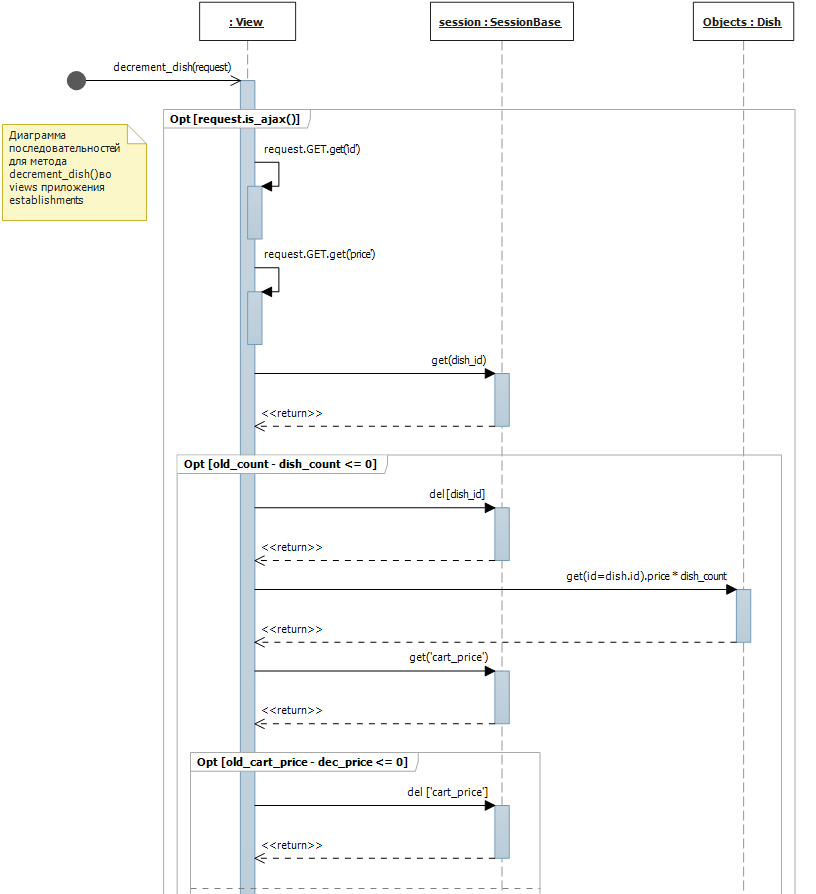


Рисунок 8 – Диаграмма последовательности increment\_dish(request) в Orders

При вызове increment\_dish(request) происходит проверка заголовка (на предмет поддержки JavaScript). Если проверка прошла успешно, то переменным dish\_id и dish\_price передаются результаты работы методов request.GET.get('id') и request.GET.get('price'). После этого происходит увеличение количества единиц заданного блюда на 1. Как? Я хз. Если проверка не прошла, возвращается сообщение с ошибкой.

На рисунке 9 изображена диаграмма последовательности для метода decrement\_dish(request) во views приложения Orders.



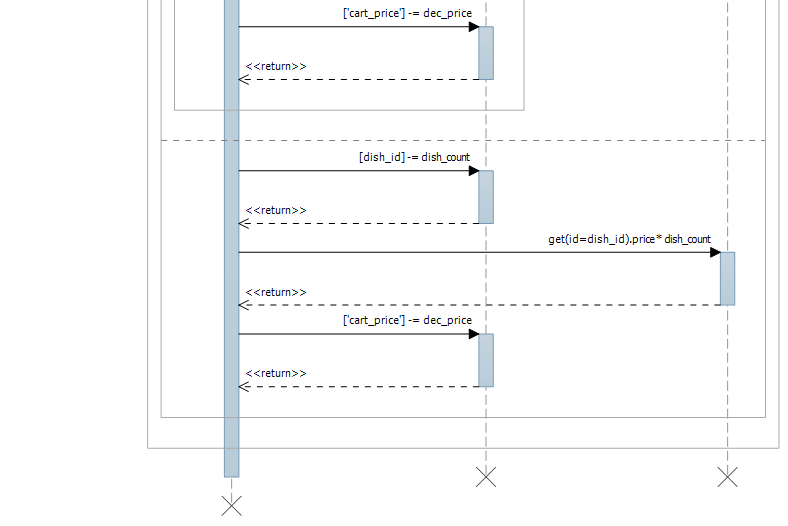


Рисунок 9 – Диаграмма последовательности decrement\_dish(request) в Orders

На рисунке 10 изображена диаграмма последовательности для метода get\_context\_data(self, \*\*kwargs) класса EmployeePage во views приложения Employees.

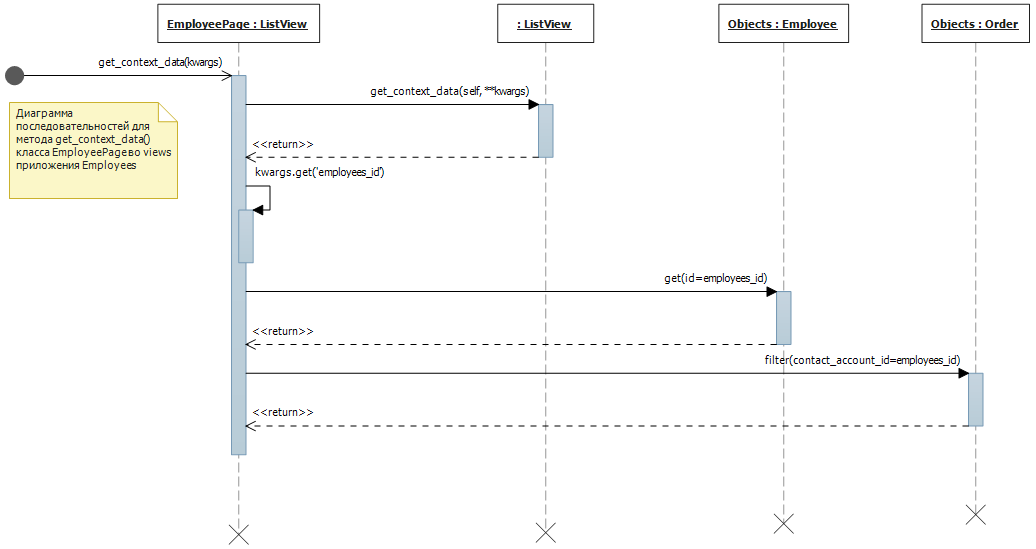
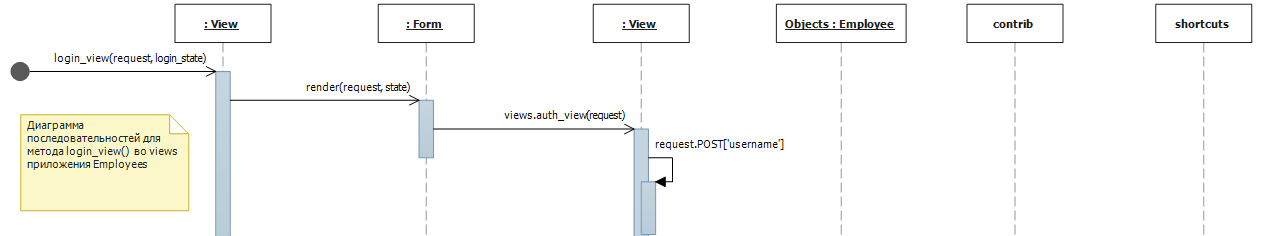


Рисунок 10 – Диаграмма последовательности get\_context\_data(self, \*\*kwargs) в Employees

При вызове get\_context\_data(self, \*\*kwargs) происходит создание нового context, которому передается результат вызова super().get\_context\_data(\*\*kwargs). После этого задается параметр emloyees\_id=self.kwargs.get(emloyees\_id'). Далее в context записываются параметры: результат get() запроса объекта Employee, по параметру id=employees\_id и фильтрация списка заказов по contact\_account\_id=employees\_id.

На рисунке 11 изображена диаграмма последовательности для метода login\_view(request, login\_state) во views приложения Emloyees.



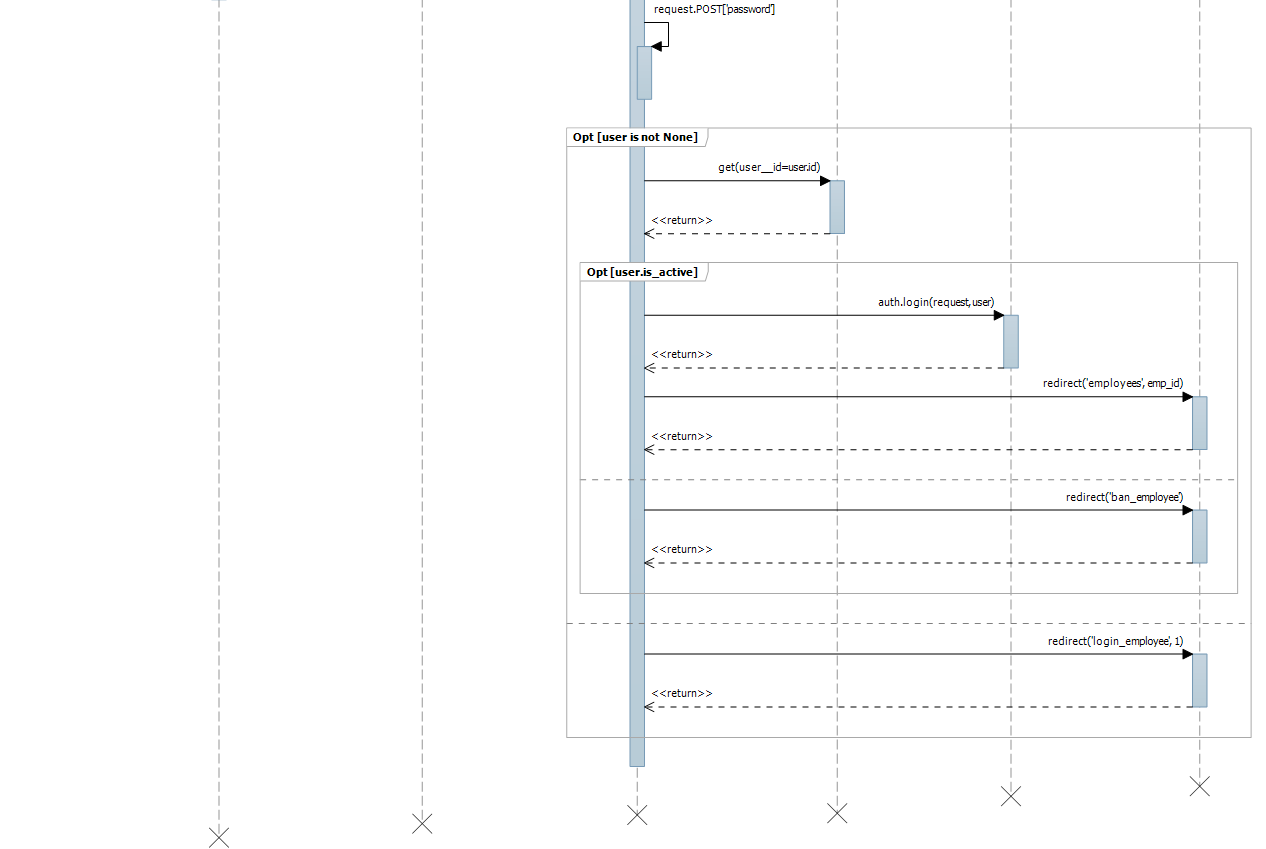


Рисунок 11 – Диаграмма последовательности login\_view(request, login\_state)в Employees

При вызове login\_view(request, login\_state)происходит проверка значения переменной состояния. Результат этой проверки влияет на отображение дополнительной информации на форме авторизации. После этого запрос с параметром передается на форме, имеющую action="{% url 'after\_login' %}" и method="post". Форма, получив от пользователя связку логин/пароль, передает ее в auth\_view(request), где с помощью двойной проверки проверяется, существует ли такой пользователь и не является ли он неактивным. Если обе проверки прошли успешно, то происходит перенаправление на страницу заказов сотрудника. Если пользователь неактивен, происходит перенаправление на страницу с уведомлением об этом. Если данные были введены неверно, происходит перенаправление на страницу авторизации с параметром state=1. В результате отображается страница авторизации с уведомление, что данные были указаны неверно.