

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт: Институт Кибернетики

Направление подготовки (специальность): Информатика и вычислительная техника

Кафедра: Оптимизации Систем Управления

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту**

по дисциплине «Технологии разработки программного обеспечения»

на тему «Сервис для оформления заказов в ресторанах “Wanna eat”»

Выполнили студенты гр. 8ВМ43

К.А. Костин
К.А. Гаврилов
С.А. Агеева
М.Р. Рафиков

Дата сдачи пояснительной записки преподавателю

_____ 20__ г.

Руководитель: ассистент каф. ОСУ

И.А. Заикин

(Оценка руководителя)

(Подпись)

_____ 20__ г.
(Дата проверки)

Курсовой проект студенты К.А. Костин, К.А. Гаврилов, С.А. Агеева, М.Р. Рафиков
выполнили и защитили
(Ф.И.О.)
с оценкой _____.

_____ 20__ г.
(дата защиты)

Томск 2014 г.

Оглавление

Введение	4
1 Требования к программе	6
1.1 Назначение программы	6
1.2 Область применения	6
1.3 Задачи, решаемые программой.....	6
1.4 Функциональные требования	6
1.5 Требования к надежности	8
1.6 Требования к эргономике и технической эстетике	8
1.7 Требования к программной документации	9
1.8 Стадии и этапы разработки	9
1.9 Диаграмма вариантов использования	10
1.10 Детализация вариантов использования	13
1.10.1 Вариант использования «Заказать столик».....	13
1.10.2 Вариант использования «Просмотреть корзину заказов»	14
1.10.3 Вариант использования «Просмотреть список блюд».....	17
1.10.4 Вариант использования «Заказать самовывоз»	19
2 Анализ	21
2.1 Диаграмма классов анализа	21
2.1.1 Заведения	21
2.1.2 Блюда.....	23
2.1.3 Заказы.....	24
2.1.4 Управление заведений.....	26
2.2 Диаграмма состояний	26
3 Проектирование.....	29
3.1 Проектные классы.....	29
3.2 Диаграмма пакетов системы	32
3.3 Диаграммы последовательностей для операций проектных классов	34
4 Реализация.....	43
4.1 Тестирование	43
4.1.1 Модульные тесты.....	43

4.1.2 Интеграционные тесты	44
4.1.3 Построение и выполнение тестов	45
4.1.4 Покрытие кода.....	46
4.1.5 Запуск приложения для тестирования	47
4.2 Непрерывная интеграция	48
5 Документация	52
5.1 Назначение программы	52
5.2 Условия запуска программы.....	52
5.3 Выполнение программы.....	53
5.3.1 Авторизация работника заведения.....	53
5.3.2 Просмотр списка заказов работником заведения	54
5.3.3 Выбор блюд по категории.....	54
5.3.4 Добавление блюда в корзину.....	55
5.3.5 Работа с корзиной	55
5.3.6 Формирование заказов	56
5.3.7 Оформление заказа	57
5.3.8 Просмотр заказов клиентом.....	59
5.4 Сообщения программы.....	59
Заключение	62
Список использованных источников	63
Приложение А	65
Приложение Б	67
Приложение В.....	68
Приложение Г	69

Введение

В настоящее время существует множество сервисов, позволяющих пользователям делать заказы различных услуг и продуктов через интернет. При этом широкое распространение мобильных устройств привело к созданию универсальных web приложений, которые обладают удобством использования как при работе с помощью персональных компьютеров, так и с помощью мобильных устройств. Создание универсальных сайтов, позволяющих предоставить пользователю самый широкий спектр услуг, является одним из актуальных направлений современной web разработки. Одной из сфер, предоставляющих подобные услуги, являются фирмы – владельцы ресторанов, кафе, баров и прочих заведений города, предоставляющих услуги заказа блюд. В настоящее время хорошим тоном каждого из таких ресторанов является наличие собственного web сайта, позволяющего делать заказы в этом заведении или других его филиалах [1].

Наиболее универсальным путем для пользователей является объединение меню нескольких заведений разных городов в одном приложении. При этом пользователи могут быстрым и удобным способом без предварительной регистрации совершать заказы в представленных заведениях.

Поэтому целью данного курсового проекта является создание приложения для оформления заказов различных заведений нескольких городов. При этом заказы могут быть заказами по бронированию столика, доставке блюд или заказами, подразумевающими вывоз блюд самим заказчиком.

Для создания web приложения данного проекта используется платформа Django на основе языка программирования Python. Выбор данной платформы заключается в том, что она позволяет создавать web ресурсы значительно быстрее, чем с помощью других платформ, например, на основе php, а также позволяет не создавать лишних элементов, не подстраиваться под заданные рамки, которые присутствуют при использовании различных CMS.

Повышенная безопасность, надежность исходных кодов, написанных на python, снижает вероятность взлома и уязвимости от различного рода атак. Использование в качестве шаблона проектирования MVC, позволяет эффективно реализовать и упростить работу по разработке сайта. Открытость платформы позволяет не тратить ресурсы на покупку лицензий [2].

Для написания кода приложения используется среда разработки на языке python PyCharm Community Edition, которая также распространяется свободно.

1 Требования к программе

1.1 Назначение программы

Web приложение “Wanna eat” предназначено для оформления заказов в различных заведениях, предоставляющих услуги заказов блюд в различных городах.

1.2 Область применения

Данный программный продукт является сервисом, предназначенным для организаций, владеющих сетью кафе, ресторанов, баров, расположенных в различных городах, а также для организаций, предоставляющих актуальную информацию о ресторанных заведениях различных городов.

1.3 Задачи, решаемые программой

1. Язык реализации – Python с интерпретатором версии не ниже 3.4;
2. Фреймворк Django версии не ниже 1.7;
3. Основной язык программы – русский;
4. Система должна производить валидацию полей не более, чем в течении трех секунд;
5. Система должна производить авторизацию пользователя не более, чем в течении трех секунд;
6. Функциональная система должна разрабатываться и функционировать в рамках существующего законодательства Российской Федерации.

1.4 Функциональные требования

1. Программа “Wanna eat” (далее система) должна предоставлять полный список заведений города, которые добавлены в данную систему, с отображением логотипа заведения и описания заведения;
2. Система должна отображать список заведений отфильтрованных по выбранному городу;

3. Система должна предоставить полный список блюд заведения с отображением изображения блюда, состава блюда, цены порции блюда и веса блюда;
4. Система должна отображать отфильтрованный список блюд заведения в соответствии с выбранной категорией;
5. Система должна предоставлять подробное описание блюда, которое должно включать в себя: изображение блюда, описание блюда, состав блюда, цену блюда, вес блюда;
6. Система должна предоставлять возможность добавить порцию блюда в корзину заказов из меню заведения (при просмотре списка блюд) и из меню просмотра подробной информации о блюде;
7. Система должна подсчитывать общую сумму стоимости порций блюд, добавленных в корзину заказа;
8. Система должна предоставлять возможность просмотра корзины заказов;
9. Система должна предоставлять возможность уменьшения и увеличения количества порций блюд при просмотре корзины заказов;
10. Система должна предоставлять возможность удаления всех порций блюда из корзины заказов;
11. Система должна сортировать блюда различных заведений в отдельные заказы перед их оформлением;
12. Система должна предоставлять пользователю возможность выбора вида заказа (заказ столика, самовывоз, доставка);
13. Система должна предоставлять форму для ввода необходимых для оформления заказа данных;
14. Система должна сохранять заказ после его оформления;
15. Система должна предоставлять возможность аутентификации пользователя-сотрудника заведения;

16. Система должна предоставлять возможность авторизации пользователя - сотрудника заведения;
17. Система должна предоставлять возможность просмотра авторизованному пользователю-сотруднику заведения заказов, сделанных в заведении данной программной системы;
18. Система должна предоставлять возможность принятия заказа на исполнение и отмены заказа авторизованному пользователю-сотруднику заведения.

1.5 Требования к надежности

Работа системы должна гарантировать средний аптайм – 99 % в год. Для этого система должна быть развернута на облачном сервисе, предоставляющем платформу как услугу, либо на сервере организации-клиента под управлением последней стабильной версии Windows Server или серверной версии операционной системы Linux.

При вводе некорректной информации программная система должна выдать предупреждающее сообщение о некорректности введенных данных и предложить повторить ввод.

Восстановление системы после критических ошибок, приводящих к неработоспособности системы должно происходить в течение суток после момента возникновения ошибки.

1.6 Требования к эргономике и технической эстетике

Взаимодействие пользователя и программы осуществляется посредством графического интерфейса клиентской программной системы – браузера. Программа должна иметь элементы навигации. Программа должна быть выполнена в едином стиле, иметь одинаковое расположение основных элементов навигации. Программа должна поддерживать горячие клавиши подтверждения ввода – Enter и закрытия подсказок и всплывающих окон – Escape.

При вводе некорректных данных, должны выдаваться сообщения на основном языке программной системы о неправильном вводе данных и предложение ввести данные заново.

1.7 Требования к программной документации

Документация представлена в виде отчета по курсовой работе, который состоит из следующих частей:

1. Титульный лист, номинальный объем – 1 страница;
2. Содержание, номинальный объем – 1 страница;
3. Введение, номинальный объем – 1 страница;
4. Требования к программе, номинальный объем – 8 страниц;
5. Анализ, номинальный объем – 8 страниц;
6. Проектирование, номинальный объем – 8 страниц;
7. Реализация, номинальный объем – 8 страниц;
8. Документация, номинальный объем – 8 страниц;
9. Заключение, номинальный объем – 1 страница;
10. Список использованных источников, номинальный объем – 1 страница.

1.8 Стадии и этапы разработки

Сроки разработки: с 1 сентября по 22 декабря 2014 г.

Основные этапы разработки программы:

1. Описание требований к системе;
2. Выявление классов. Построение и описание диаграммы классов анализа;
3. Построение и описание диаграммы состояний;
4. Построение и описание проектных классов;
5. Построение и описание диаграмм последовательности для операций проектных классов;
6. Построение и описание диаграммы пакетов;
7. Разработка программы;

8. Модульное тестирование;
9. Настройка системы непрерывной интеграции.

1.9 Диаграмма вариантов использования

Диаграмму вариантов использования данной программной системы можно разделить на 3 части, которые представлены на Рис. 1 – Рис. 3. Цельный рисунок диаграммы вариантов использования представлен в приложении Б.

На Рис. 1 представлена часть диаграммы вариантов использования программной системы, описывающая действия по просмотру информации на сайте, добавлению блюд в корзину и просмотре заказов для актера «Клиент».

На Рис. 2 представлена часть диаграммы вариантов использования программной системы, описывающая действия по просмотру корзины заказов и оформлению заказов для актера «Клиент».

На Рис. 3 представлена часть диаграммы вариантов использования программной системы, описывающая действия актера «Сотрудник заведения» по просмотру заказов заведения и их принятию к исполнению или отмене, а также действия актера «Время», удаляющего устаревшие заказы из базы данных.

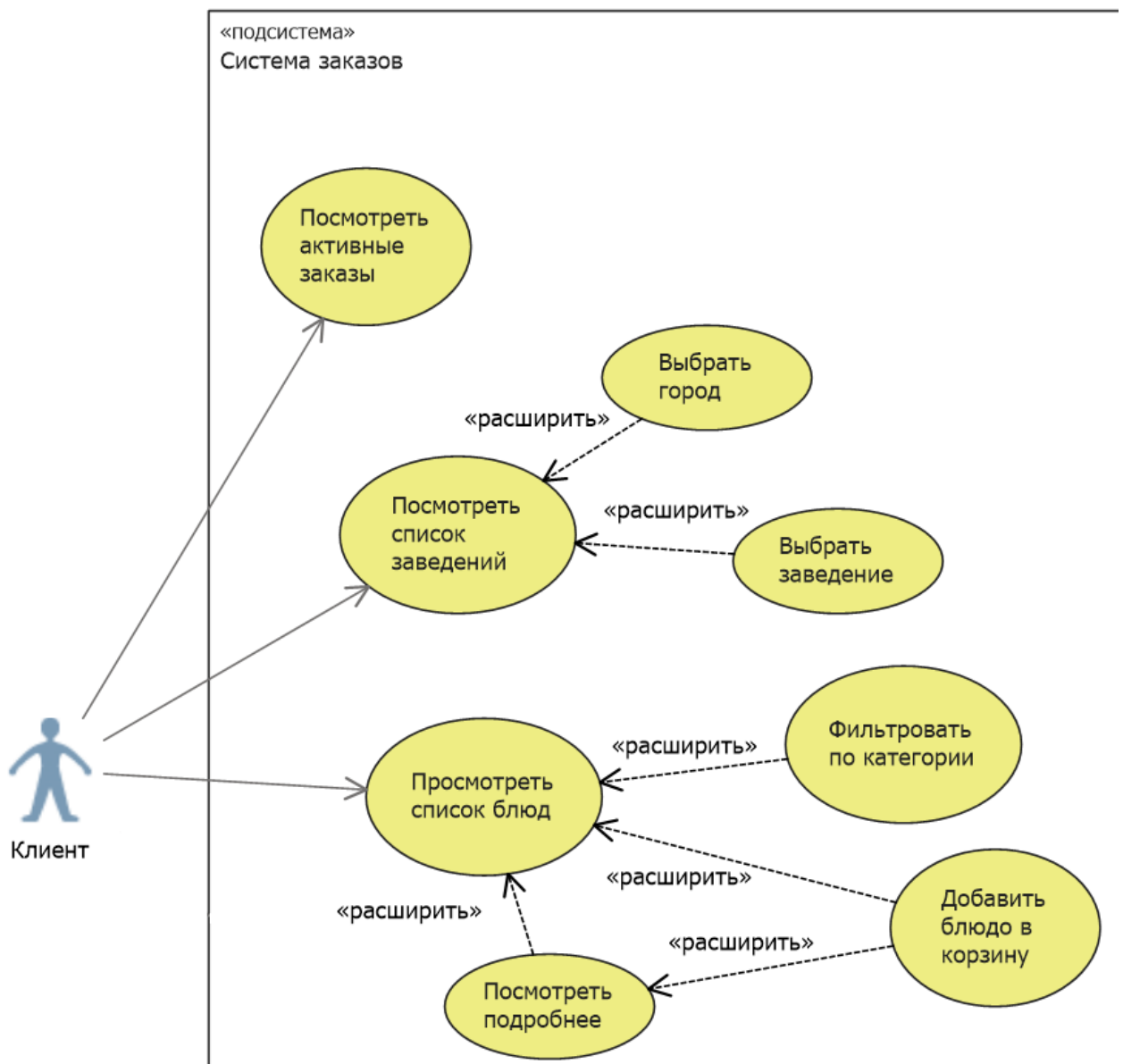


Рис. 1. Часть диаграммы вариантов использования программной системы (1)

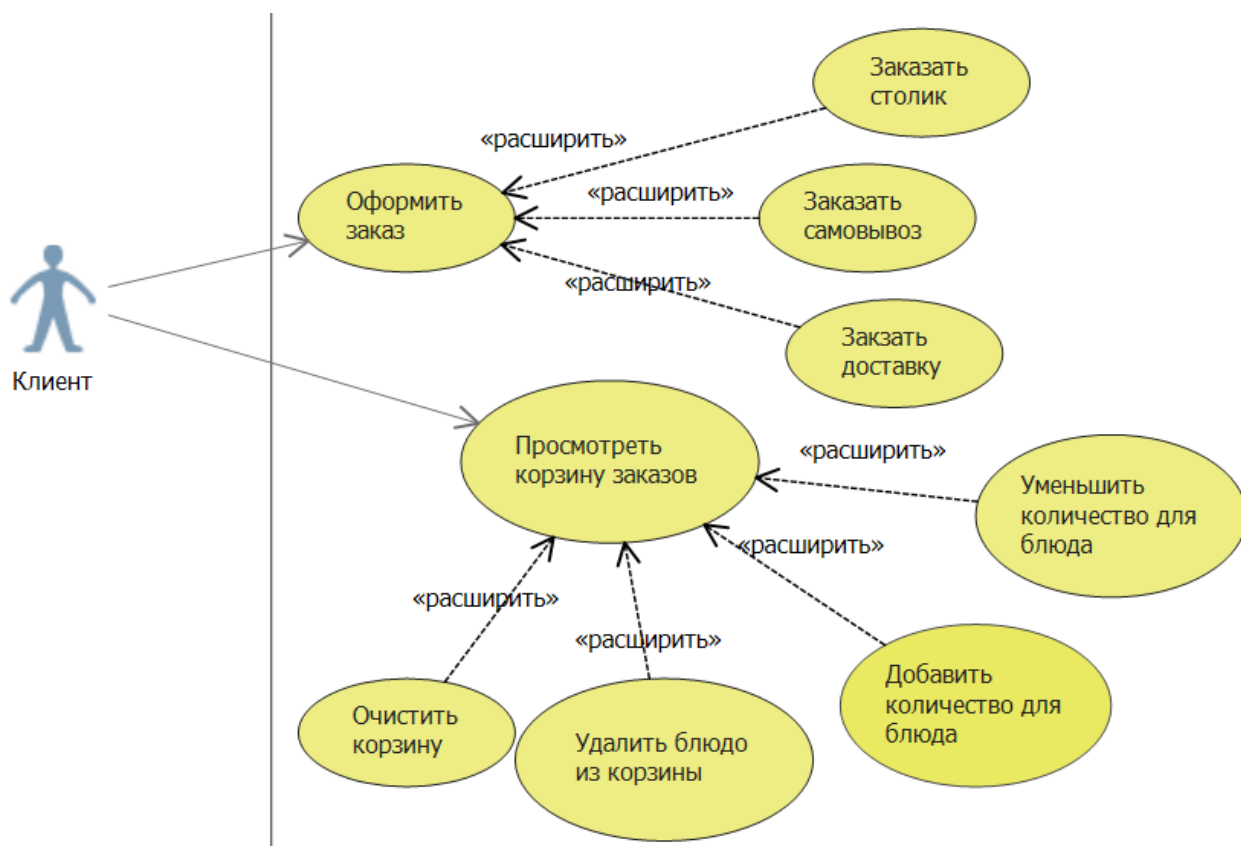


Рис. 2. Часть диаграммы вариантов использования программной системы (2)



Рис. 3. Часть диаграммы вариантов использования программной системы (3)

1.10 Детализация вариантов использования

Ниже представлены спецификации некоторых вариантов использования, изображенных на Рис. 1 – Рис. 3 [3, с. 101-103].

1.10.1 Вариант использования «Заказать столик»

Вариант использования: Заказать столик
Краткое описание: Клиент бронирует столик в заведении.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Выбрано заведение.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает опцию «Заказать столик»; 2. Клиент вводит номер своего мобильного телефона; 3. Клиент выбирает дату и время бронирования столика; 4. Клиент выбирает филиал заведения, зал заведения и бронируемый столик; 5. Клиент подтверждает введенные данные; 6. Система бронирует столик; 7. Система возвращает Клиента к странице просмотра меню заведения
Постусловия: 1. Заказ бронирования столика оформлен.
Альтернативные потоки: 1. Отмена заказа бронирования столика; 2. Отсутствие свободных столиков; 3. Неуспешное бронирование столика.

<u>Альтернативный поток: Заказать столик: Отмена заказа столика</u>
Краткое описание: Отмена заказа столика в процессе оформления заказа.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Выбрана опция отмены заказа.

<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Альтернативный поток начинается на любом шаге (до 6-го) основного потока; 2. Система возвращает клиента на предыдущую страницу.
<p>Постусловия:</p> <ol style="list-style-type: none"> 1. Заказ бронирования не оформлен.

<u>Альтернативный поток: Заказать столик: Отсутствие свободных мест</u>
<p>Краткое описание:</p> <p>Отсутствие свободных столиков в филиале заведения.</p>
Главные актеры: Клиент.
Второстепенные актеры: нет.
<p>Предусловия:</p> <ol style="list-style-type: none"> 1. Выбран филиал заведения;
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Альтернативный поток начинается на шаге 4 основного потока; 2. Система оповещает Клиента о невозможности выбора столика;
<p>Постусловия:</p> <ol style="list-style-type: none"> 1. Заказ бронирования не оформлен.

<u>Альтернативный поток: Заказать столик: Неудачное бронирование столика</u>
<p>Краткое описание:</p> <p>Невозможность бронирования столика по причинам ошибок передачи данных.</p>
Главные актеры: Клиент.
Второстепенные актеры: нет.
<p>Предусловия:</p> <ol style="list-style-type: none"> 1. Введены все данные, необходимые для оформления заказа бронирования столика; 2. Клиент подтвердил введенные данные.
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Альтернативный поток начинается на шаге 6 основного потока; 2. Система сообщает Клиенту о неудачном бронировании столика;
<p>Постусловия:</p> <ol style="list-style-type: none"> 1. Заказ бронирования не оформлен.

1.10.2 Вариант использования «Просмотреть корзину заказов»

Вариант использования: Просмотреть корзину заказов
Краткое описание:

Просмотр блюд заказов и возможность произвести действия, расширяющие просмотр блюд заказов.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. В корзине заказов присутствует хотя бы один пункт.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает опцию просмотра корзины заказов; 2. Для каждого пункта в корзине: 2.2 Система выводит название заведения и сумму блюд заказ в этом заведении; 3. Система выводит общую стоимость блюд в корзине; Точка расширения: «Очистить корзину»; Точка расширения: «Удалить блюдо из корзины»; Точка расширения: «Добавить количество для блюда»; Точка расширения: «Уменьшить количество для блюда».
Постусловия: нет.

<u>Расширяющий вариант использования: Очистить корзину</u>
Краткое описание: Удаление всех блюд из корзины заказов.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Система отображает корзину заказов.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает опцию очистки корзины заказов; 2. Система удаляет все блюда из корзины заказов; 3. Система уменьшает цену корзины заказов до нуля.
Постусловия: 1. Все блюда из корзины заказов удалены.

<u>Расширяющий вариант использования: Удалить блюдо из корзины</u>
Краткое описание: Удаление выбранного блюда из корзины заказов.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Система отображает корзину заказов.

<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Вариант использования начинается, когда Клиент выбирает опцию удаления блюда из корзины; 2. Система удаляет все порции выбранного блюда из корзины. 3. Система рассчитывает общую цену корзины заказов, уменьшая её на величину, равную произведению цены выбранного блюда на количество порций блюда.
<p>Постусловия:</p> <ol style="list-style-type: none"> 1. Выбранное блюдо удалено из корзины заказов.
<p>Альтернативные потоки:</p> <ol style="list-style-type: none"> 1. Очистка корзины заказов.

<p><u>Расширяющий вариант использования: Добавить количество для блюда</u></p>
<p>Краткое описание: Увеличение количество порций блюда на единицу.</p>
<p>Главные актеры: Клиент.</p>
<p>Второстепенные актеры: нет.</p>
<p>Предусловия:</p> <ol style="list-style-type: none"> 1. Система отображает корзину заказов.
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Вариант использования начинается, когда Клиент выбирает опцию увеличения количества порций для блюда; 2. Система увеличивает количество порций для блюда; 3. Система рассчитывает общую цену корзины заказов, увеличивая её на величину, равную цене выбранного блюда.
<p>Постусловия:</p> <ol style="list-style-type: none"> 1. Количество порций блюда увеличено на единицу.

<p><u>Альтернативный поток: Уменьшить количество для блюда: Удаление блюда из корзины</u></p>
<p>Краткое описание: Удаление блюда из корзины при уменьшении количества порций до нуля.</p>
<p>Главные актеры: Клиент.</p>
<p>Второстепенные актеры: нет.</p>
<p>Предусловия:</p> <ol style="list-style-type: none"> 1. Количество единиц блюда в корзине заказов становится равным нулю.
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Альтернативный поток начинается на шаге 2 основного потока;

2. Система удаляет блюдо из корзины заказов; 3. Система рассчитывает общую цену корзины заказов, уменьшая её на величину, равную цене выбранного блюда.
Постусловия: 1. Выбранное блюдо удалено из корзины заказов.
Альтернативные потоки: 1. Очистка корзины заказов.

<u>Альтернативный поток: Удаление блюда из корзины/ Удалить блюдо из корзины: Очистка корзины заказов</u>
Краткое описание: Очистка корзины заказов при уменьшении общей цены корзины заказов до нуля.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Цена корзины заказов становится равной нулю.
Основной поток: 1. Альтернативный поток начинается на шаге 3 основного потока. 2. Система уменьшает цену корзины заказов до нуля.
Постусловия: 1. Все блюда из корзины заказов удалены.

1.10.3 Вариант использования «Просмотреть список блюд»

Вариант использования: Просмотреть список блюд
Краткое описание: Просмотр списка блюд заведения и возможность произвести действия, расширяющие просмотр блюд заведения.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: нет.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает опцию просмотра меню заведения; 2. Для всех блюд заведения система вывод на экран: 2.2 Изображение блюда, цену блюда, вес блюда. Точка расширения: «Посмотреть подробнее»; Точка расширения: «Фильтровать по категории»; Точка расширения: «Добавить блюдо в корзину».
Постусловия: нет.

<u>Расширяющий вариант использования: Посмотреть подробнее</u>
Краткое описание: Просмотр подробной информации о блюде.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Отображение страницы меню заведения.
Основной поток: 1. Вариант использования начинается, когда Клиент выберет опцию просмотра подробной информации о блюде; 2. Система вывод на экран изображение блюда, цену блюда, вес блюда, состав блюда, описание блюда; Точка расширения: «Добавить блюдо в корзину».
Постусловия: нет.

<u>Расширяющий вариант использования: Фильтровать по категории.</u>
Краткое описание: Просмотр блюд заведения только определенной категории.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Отображение страницы меню заведения.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает определённую категорию на странице меню заведения; 2. Для каждого блюда заведения, имеющего выбранную категории система вывод на экран: 2.2Изображение блюда, цену блюда, вес блюда.
Постусловия: нет.

<u>Расширяющий вариант использования: Добавить блюдо в корзину.</u>
Краткое описание: Добавление выбранного блюда в корзину заказов.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. Отображение страницы меню заведения; 2. Отображение страницы детализации блюда.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает опцию добавления блюда в корзину заказов;

2. Если в корзине заказов уже присутствует выбранное блюдо; 2.2 Количество порций блюда в корзине заказов увеличивается на единицу; 3. Если в корзине заказов отсутствует выбранное блюдо; 3.3 Система добавляет выбранное блюдо в корзину заказов; 4. Система рассчитывает новую цену корзины заказов, увеличивая её на величину, равную цене выбранного блюда; 5. Система отображает новую цену корзины заказов;
Постусловия: 1. Выбранное блюдо добавлено в корзину заказов.

1.10.4 Вариант использования «Заказать самовывоз»

Вариант использования: Заказать самовывоз
Краткое описание: Клиент заказывает самовывоз заказа в заведении.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 1. В корзине заказов есть хотя бы одно блюдо; 2. Система отображает корзину заказов.
Основной поток: 1. Вариант использования начинается, когда Клиент выбирает опцию заказа самовывоза; 2. Клиент вводит номер своего мобильного телефона; 3. Клиент выбирает дату и время самовывоза; 4. Клиент выбирает филиал заведения; 5. Клиент подтверждает введенные данные; 6. Система оформляет заказ самовывоза; 7. Система возвращает Клиента к странице просмотра корзины заказов.
Постусловия: 1. Заказ самовывоза оформлен.
Альтернативные потоки: 1. Отмена заказа самовывоза; 2. Неудачное оформление заказа самовывоза.

<u>Альтернативный поток: Заказать самовывоз: Отмена заказа самовывоза</u>
Краткое описание: Отмена заказа самовывоза в процессе оформления заказа.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия:

1. Выбрана опция отмены заказа.
Основной поток: 3. Альтернативный поток начинается на любом шаге (до 6-го) основного потока; 4. Система возвращает клиента на предыдущую страницу.
Постусловия: 2. Заказ самовывоза не оформлен.

<u>Альтернативный поток: Заказать самовывоз: Неудачное оформление заказа самовывоза</u>
Краткое описание: Невозможность заказа самовывоза по причинам ошибок передачи данных.
Главные актеры: Клиент.
Второстепенные актеры: нет.
Предусловия: 3. Введены все данные, необходимые для оформления заказа самовывоза; 4. Клиент подтвердил введенные данные.
Основной поток: 4. Альтернативный поток начинается на шаге 6 основного потока; 5. Система сообщает Клиенту о неудачном оформлении заказа самовывоза;
Постусловия: 2. Заказ самовывоза не оформлен.

2 Анализ

Этап анализа при разработке программного обеспечения является основным этапом при использовании тяжеловесных методологий разработки. На данном этапе полученные на шаге выявления требований функциональные требования к программной системе представляются в виде отношений между классами предметной области. Создается аналитическая модель предметной области путем выявления классов и отношений между ними. При создании аналитической модели крайне важно ограничиться лишь теми классами, которые являются частью словаря предметной области. Это поможет сделать её кратким, простым и понятным описанием поведения системы [3, с. 144-145].

Классы анализа должны представлять собой четкую абстракцию предметной области и должны проецироваться на реальные бизнес-понятия предметной области.

Данная программная система состоит из четырех логических модулей, каждый из которых содержит классы, соответствующие определенным логическим аспектам поведения системы: это классы, содержащие в себе информацию о заведениях города, необходимые для оформления заказов; классы, содержащие информацию о блюдах, предлагаемых в меню данных заведений; классы, описывающие заказ, который клиенты могут оформить в существующих заведениях; классы, описывающие работников заведений, в обязанности которых входит обработка создающихся в приложении заказов. Общая схема диаграммы классов анализа представлена в приложении В.

2.1 Диаграмма классов анализа

2.1.1 Заведения

На Рис. 4 представлена часть диаграммы классов, описывающая классы, хранящие информацию о заведениях системы.

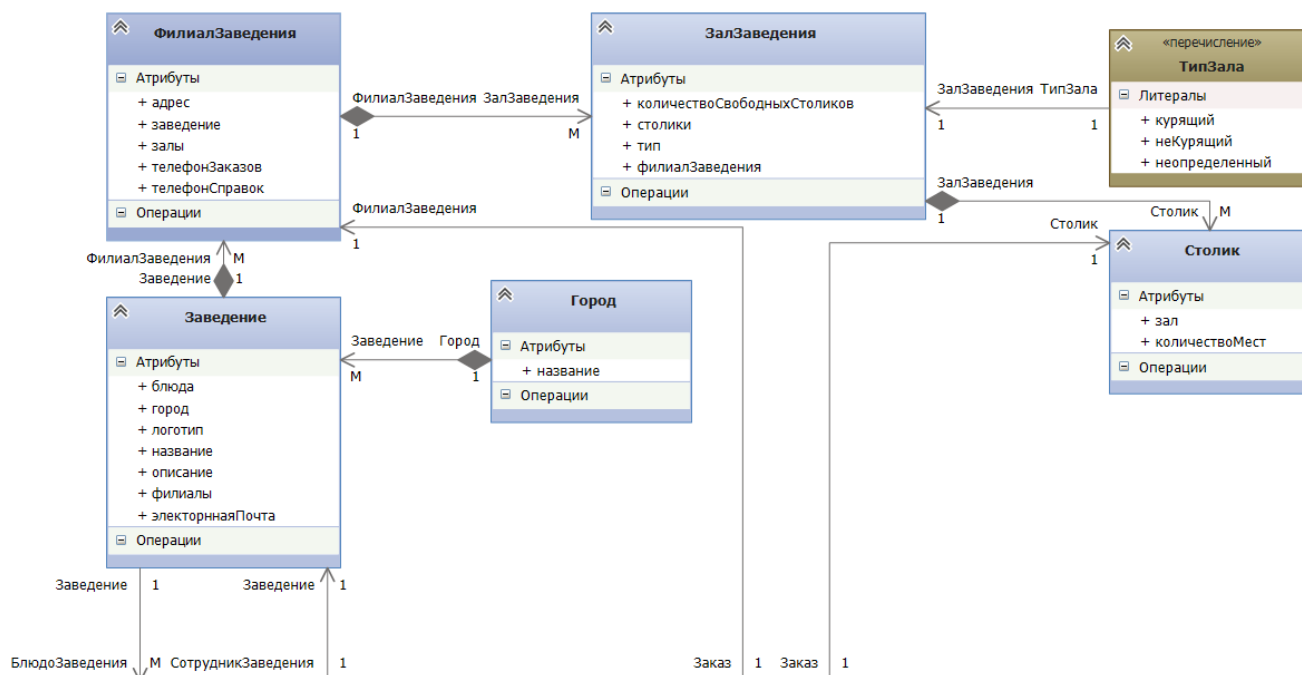


Рис. 4. Классы заведения на диаграмме классов анализа

Как видно на Рис. 4, диаграммы классов анализа, описывающие поведение заведений состоят из пяти классов и одной сущности типа перечисление.

Класс «Город» связан отношением композиции с классом «Заведение» с кратностью связи 1: М. Это означает, что класс «Город» может содержать в себе множество объектов класса «Заведение». При этом их связь является жесткой: при исчезновении объекта класса «Город» исчезают все объекты класса «Заведение».

Класс «Заведение» содержит в себе базовую информацию о заведении города. Он связан отношением композиции с классом «ФилиалЗаведения» с кратностью 1: М. Также он связан отношением ассоциации с классом «СотрудникЗаведения» с кратностью 1: 1. Это означает, что у одного заведения может быть только один сотрудник, отвечающий за обработку поступающих этому заведению заказов. Также класс «Заведение» связан отношением ассоциации с кратностью 1: М с классом «БлюдоЗаведения» (Рис. 5), который необходим для разрешения связи М: М между классами «Заведение» и «Блюдо».

Класс «ФилиалЗаведения» связан с помощью отношения композиции с классом «ЗалЗаведения» с кратностью 1: М. Также класс «Заказ» связан с классом «ФилиалЗаведения», т.к. заказ доставки и самовывоза требует указания в объекте класса «Заказ» адреса, по которому клиент может получить готовый заказ.

Класс «ЗалЗаведения» связан с помощью отношения композиции с классом «Столик» с кратностью 1: М. Вся иерархия отношений композиции между классами, описывающими заведения, представляет собой иерархическую структуру заведений города для данной программной системы. Также отношением ассоциации перечисление «ТипЗала» связано с классом «ЗалЗаведения». Данное перечисление необходимо для указания при заказе столика предпочитаемого клиентом типа зала заведения. Класс «ЗалЗаведения» имеет методы, с помощью которых реализуется поведение, необходимое для бронирования столика в заведении: забронировать столик, освободить столик, а также метод, необходимый для пересчета хранящегося в классе параметра количества свободных столиков в данном зале.

Класс «Столик» необходим для хранения информации о столике заведения. Он связан с классом «ЗалЗаведения» отношением композиции. Также класс «Заказ» (Рис. 6) связан отношением ассоциации с классом «Столик», храня тем самым информацию о столике заведения, который указан в каждом заказе на бронирование в заведении.

2.1.2 Блюда

На Рис. 5 представлена часть диаграммы классов, содержащая классы, описывающие блюда, представленные в меню различных заведений города.

Как видно на Рис. 5, классы, описывающие блюда заведения состоят из двух классов-сущностей и перечисления, описывающего категорию блюда.

Класс «Блюдо» содержит в себе информацию о блюде, а перечисление «КатегорияБлюда» связано с данным классом с помощью отношения ассоциации.

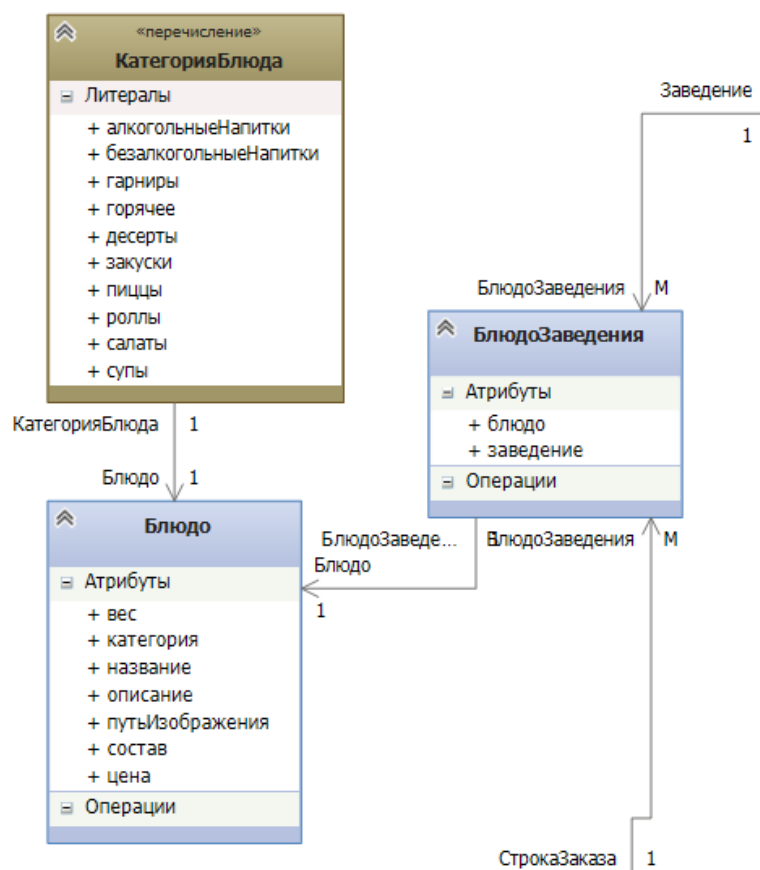


Рис. 5. Классы блюд на диаграмме классов анализа

Класс «БлюдоЗаведения» необходим, как уже говорилось выше, для разрешения связи М: М между классами «Блюдо» и «Заведение». Также класс «СтрокаЗаказа», являющийся разрешением связи М: М Классов «БлюдоЗаведения» и «КорзинаЗаказов» (Рис. 6) связан с классом «БлюдоЗаведения» отношением ассоциации с кратностью 1: М.

2.1.3 Заказы

На Рис. 6 представлены классы системы, описывающие заказы и работников заведений, обрабатывающих поступающие заказы.

Класс «СтрокаКорзиныЗаказов», как уже говорилось выше, является классом, разрешающим отношение М: М между классами «КорзинаЗаказов» и «БлюдоЗаведения». Данный класс содержит информацию о блюде, добавленном в корзину и количестве порций данного блюда. Класс «КорзинаЗаказов» содержит в себе экземпляры класса

«СтрокаКорзиныЗаказов», поэтому они связаны между собой отношением композиции с кратностью связи 1: M.

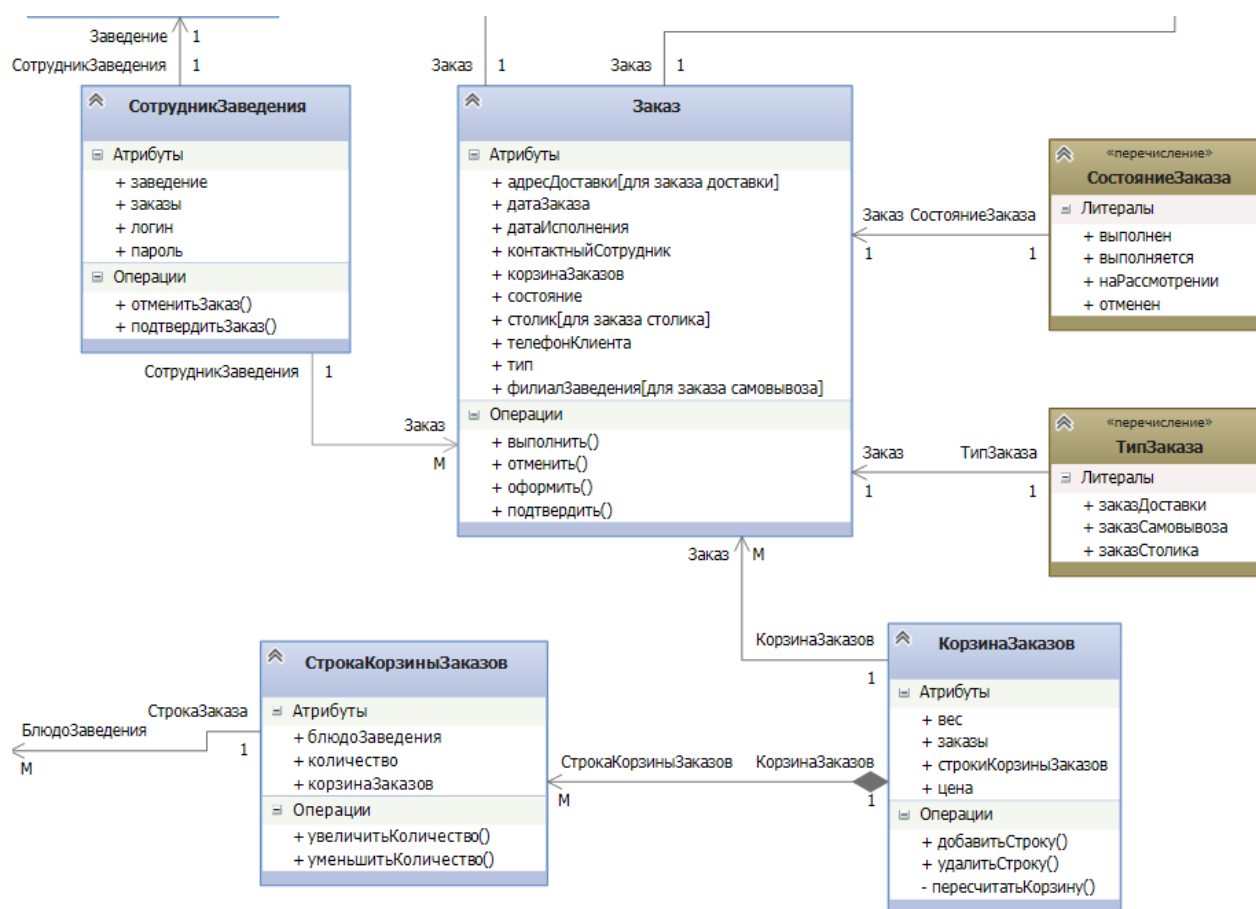


Рис. 6. Классы заказов и сотрудников заведений на диаграмме классов анализа

Класс «СтрокаКорзиныЗаказов» содержит методы, увеличивающие или уменьшающие количество порций блюда в строке корзины.

Класс «КорзинаЗаказов» содержит в себе информацию о строках, добавленных в корзину, и общих параметрах корзины: общая цена всех блюд в корзине и общий вес блюд в корзине. Данный класс связан отношением ассоциации с классом «Заказ» с кратностью 1: M. Также, класс содержит методы по созданию строк корзины заказов при добавлении нового блюда в корзину и метод пересчета общих параметров корзины после добавления или удаления блюд из корзины.

Класс «Заказ» является основным классом программной системы, содержащим в себе параметры заказа, данные клиента и связи с объектами

системы, необходимыми для формирования и отслеживания состояния заказа. Перечисления «СостояниеЗаказа» и «ТипЗаказа» связаны отношением ассоциации. Данный класс реализует свое поведение с помощью четырех методов, изменяющих его состояние: оформить, подтвердить, отменить и выполнить. Каждый из этих методов меняет состояние заказа.

2.1.4 Управление заведений

Класс «СотрудникЗаведения» представляет собой пользователя, имеющего доступ к заказам, оформленным на определенное заведение. Он может отменять и подтверждать заказы, что осуществляется с помощью методов класса. Таким образом осуществляется обработка заказа администрацией заведения, реализуя управление заведений в данной программной системе. Данный класс связан отношением ассоциации кратностью 1: 1 с классом «Заведение» и отношением ассоциации с классом «Заказ» с кратностью 1: М.

Представленные на диаграмме анализа классы главным образом предназначены для хранения информации о тех или иных объектах и являются классами-сущностями. Немногочисленные методы данных классов просты и не требуют дополнительных описаний.

2.2 Диаграмма состояний

Классом, изменяющимся динамически, в данной системе является только класс «Заказ», экземпляр которого изменяет свое состояние в процессе своего существования в данной системе.

Динамическое изменение состояния класса описывается с помощью диаграммы конечных автоматов. Данные диаграммы могут применяться как для описания жизненного цикла класса анализа, так и для описания поведения проектного класса. Построение диаграммы конечных автоматов применяется обычно на конечных этапах анализа или проектирования, что позволяет максимально подробно уточнить поведение того или иного класса [3, с. 474-475].

На Рис. 7 представлена диаграмма конечных автоматов для класса «Заказ» данной системы.



Рис. 7. Диаграмма конечных автоматов класса «Заказ»

Как видно на Рис. 7, класс «Заказ» может находиться в 4-х состояниях: на рассмотрении, выполняется, отменен и выполнен.

При оформлении заказа происходит создание экземпляра класса «Заказ», вызывается функция оформления заказа, которая автоматически переводит его в состояние на рассмотрении.

Из состояния заказа «на рассмотрении» перевести в любое другое состояние класс может экземпляр класса «РаботникЗаведения», который переводит его либо в состояние «Выполняется», либо в состояние «Отменен». Это зависит от того, смог ли клиент подтвердить то, что он сделал заказ в данном заведении. Это может быть осуществлено с помощью оставленного клиентом номера мобильного телефона, однако это никак не регламентируется в разрабатываемой системе.

Из состояния «Выполняется» заказ может быть переведен в состояние «Выполнен» автоматически, после того, как текущая дата будет больше даты исполнения заказа.

В данной системе присутствуют жесткие ограничения по изменению состояния заказа. Из состояния заказа «Отменен», если он переведен в него

сотрудником заведения, заказ может быть только удален, но не переведен в состояние «Выполняется». Для этого клиенту необходимо оформить новый заказ. Так же и из состояния «Выполняется» заказ уже не может быть отменен. Для этого необходимо связаться с клиентом по оставленному им при оформлении заказа номеру.

Из состояний «Отменен» и «Выполнен», в которые каждый из сформированных заказов рано или поздно придет, заказ может быть только удален. Это определяется датой истечения заказа, которая определяется автоматически, исходя из даты исполнения заказа, прибавляя 30 дней к значению этой даты.

Проделанные на данном этапе разработки программной системы действия подготовили основу для перехода к проектированию программной системы, ориентируясь на конкретные языки программирования, платформы и средства разработки. Находясь практически в самом начале цикла разработки программной системы, данный этап является одним из самых важных для тяжеловесных методологий разработки, потому что детальная проработка данного этапа позволяет снизить повышающуюся на дальнейших этапах жизненного цикла продукта стоимость вносимых изменений в готовую программную систему [4, с. 26-28].

3 Проектирование

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

3.1 Проектные классы

Проектные классы – это классы, описание которых настолько полно, что они могут быть реализованы [3, с. 372]. Описание формируется путем уточнения классов анализа, которое включает в себя добавлений деталей реализации. Полная диаграмма проектных классов представлена в приложении Г. На Рис. 8 представлена часть диаграммы проектных классов, реализуемых в системе. Эта часть диаграммы отображает группу классов, относящихся к «Заведениям».

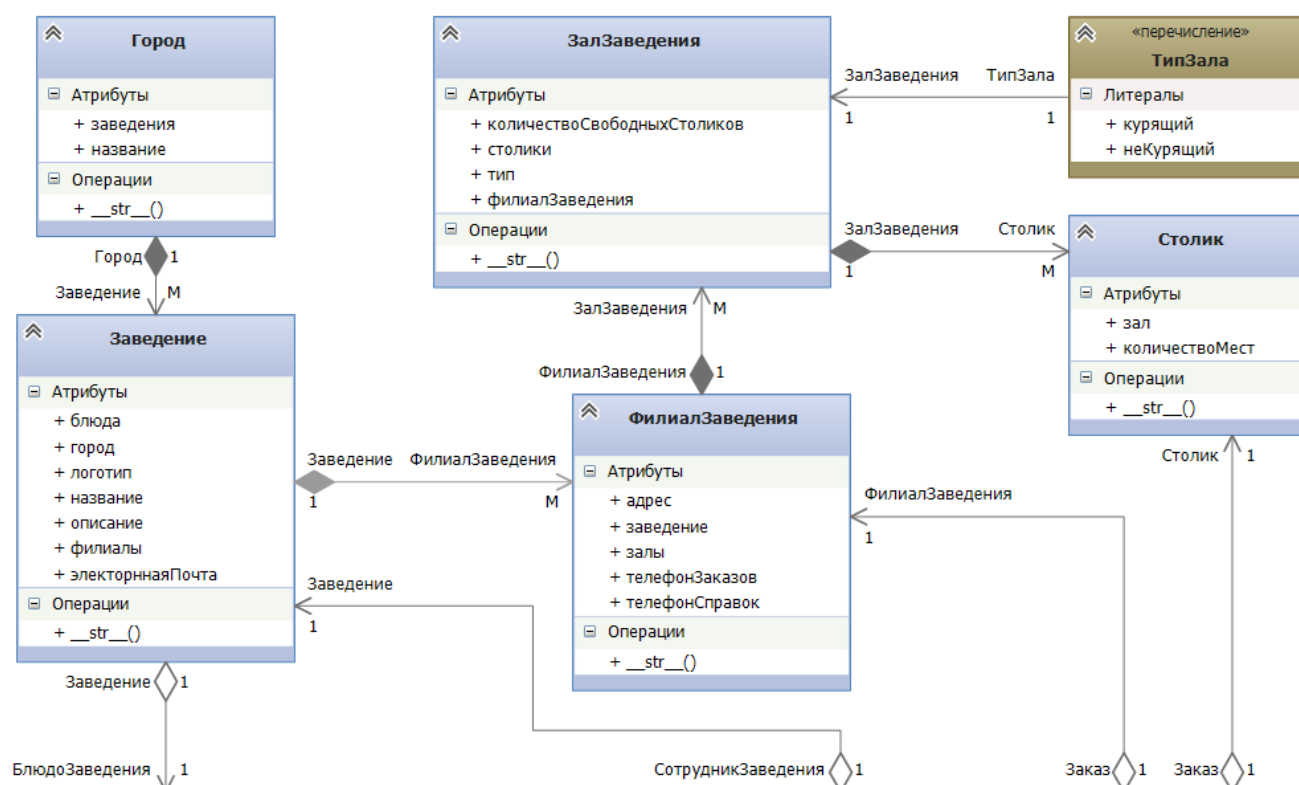


Рис. 8. Часть диаграммы проектных классов – заведения

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 Рис. 9 представлена часть диаграммы, которая отображает группу классов, относящихся к «Блюдам».

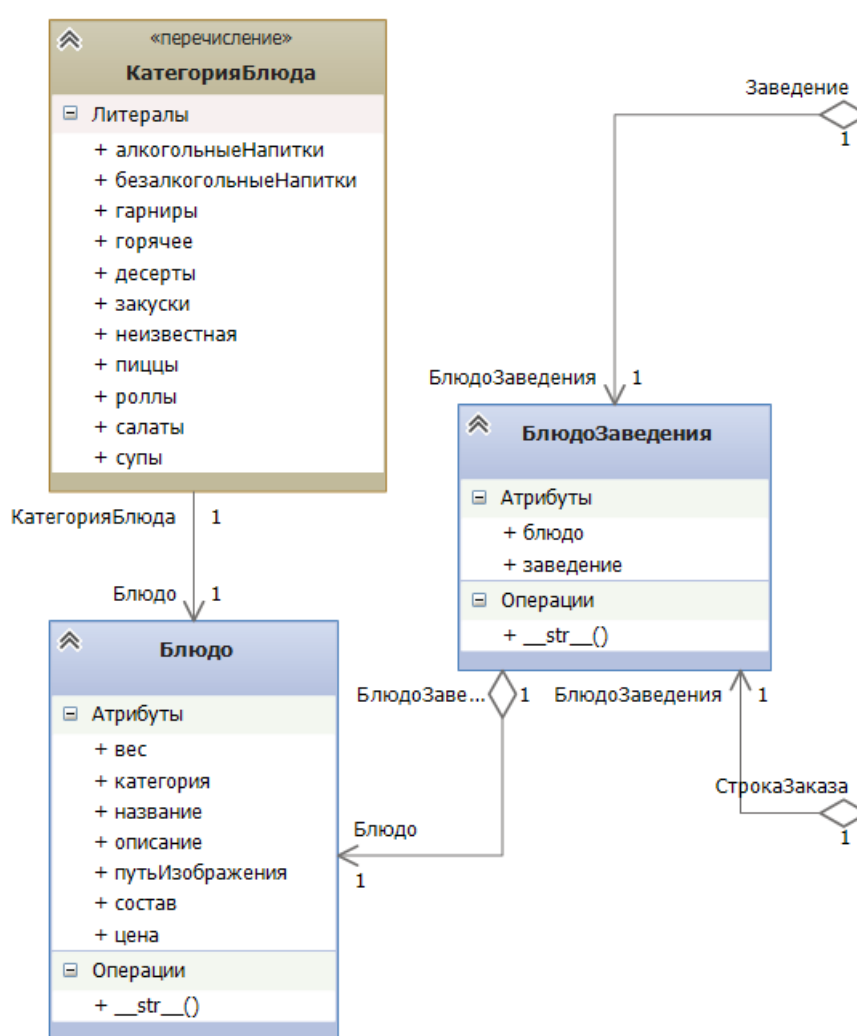


Рис. 9. Часть диаграммы проектных классов – блюда

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 Рис. 10 представлена часть диаграммы, которая отображает группу классов, относящихся к «Заказам».

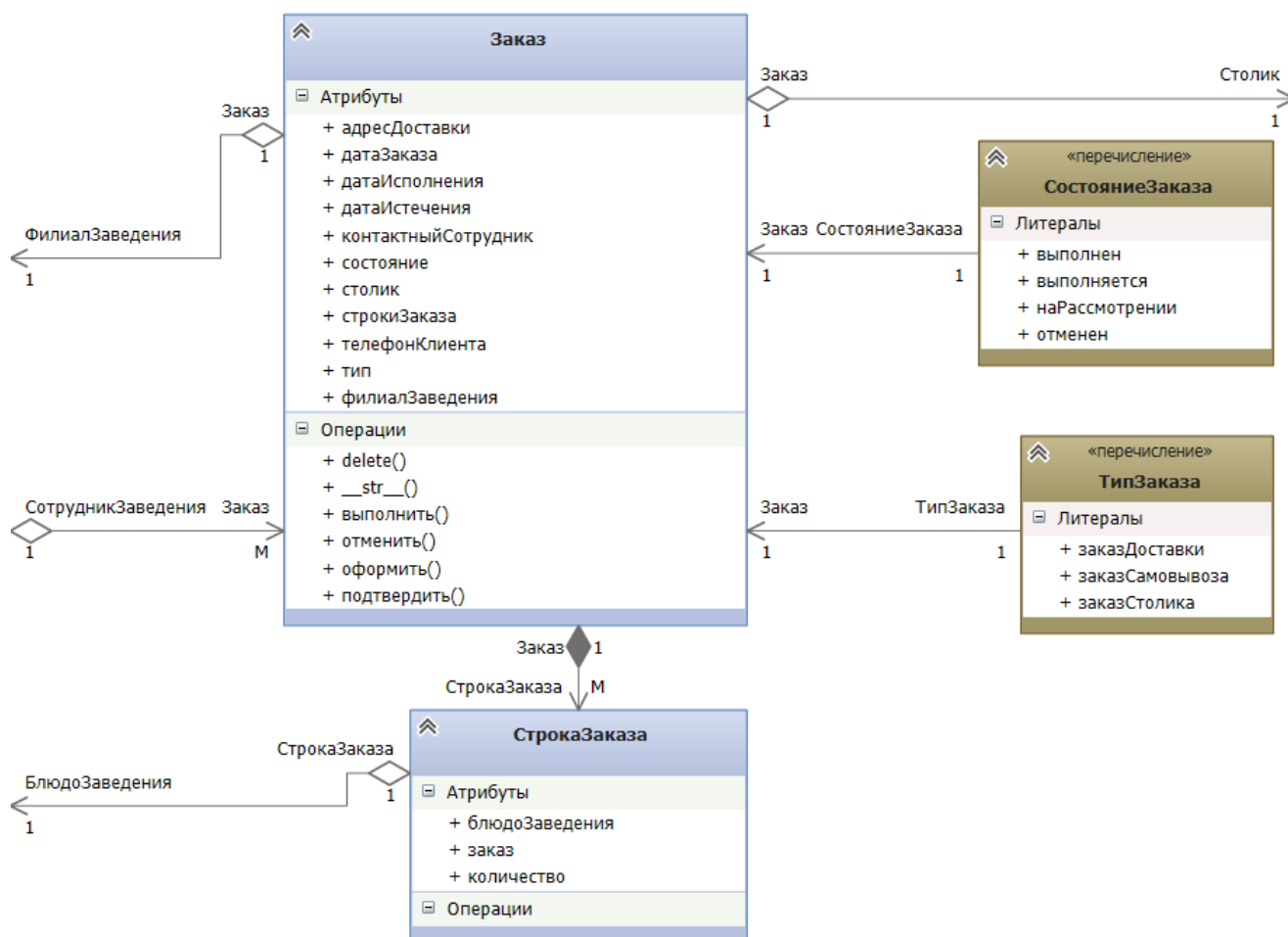


Рис. 10. Часть диаграммы проектных классов – заказы

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 Рис. 11 представлена последняя часть диаграммы, которая отображает группу классов, относящихся к «Управлению Заведений».

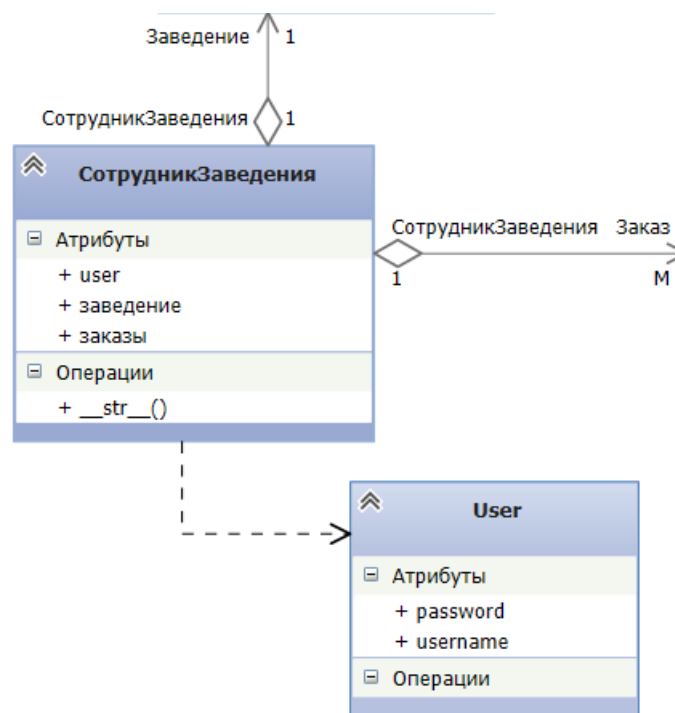


Рис. 11. Часть диаграммы проектных классов – сотрудники заведений

Здесь был добавлен класс Django, который позволяет создавать и хранить экземпляры данного класса в классе «СотрудникЗаведения», что необходимо для аутентификации и авторизации пользователей – сотрудников заведения. Экземпляры данного класса могут входить в систему под аккаунтом, который содержится в классе User, для того, чтобы управлять заказами, оформленными на данное заведение.

3.2 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 [5].

На Рис. 12 представлена диаграмма пакетов. Она состоит из 3 базовых пакетов, показывающих разделение отображения данных, описания моделей данных программы и хранения данных.

Пакет «Пользовательский интерфейс» содержит пакеты «Static», где хранятся статические ресурсы программной системы, такие как файлы

стилей CSS, файлы скриптов JS, файлы изображений. От данного пакета зависит пакет «Templates», который использует статические данные для отображения интерфейса пользователю.

Пакет «Модели» содержит классы моделей системы, распределенные по четырем пакетам: «Заведения», «Блюда», «УправлениеЗаведений» и «Заказы».

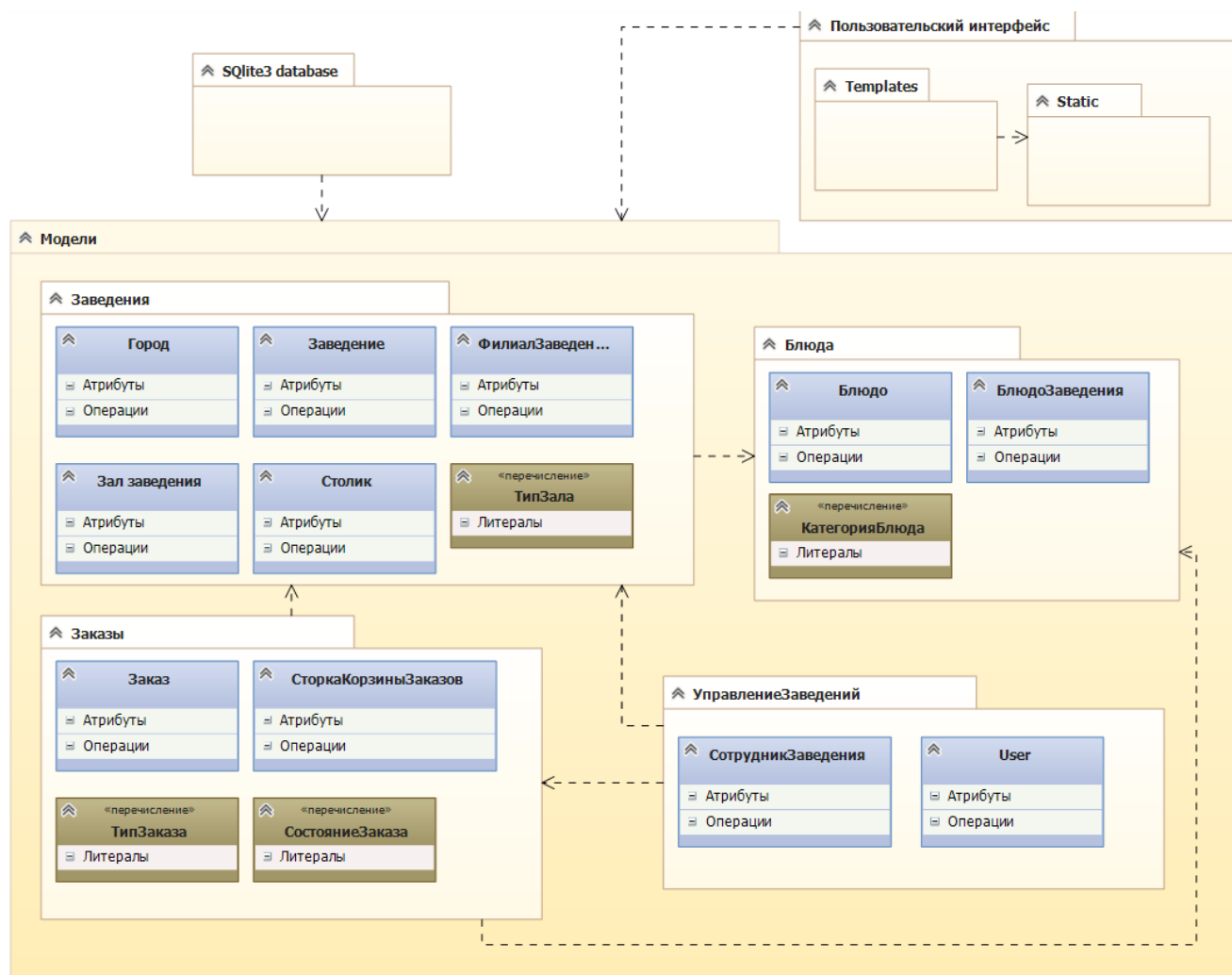


Рис. 12. Диаграмма пакетов программной системы

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».

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 SQLite3 database содержит в себе базу данных приложения.

3.3 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 [6]. 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 [7].

В ходе выполнения курсового проекта были спроектированы диаграммы последовательностей для нескольких функций. На Рис. 13 изображена диаграмма последовательности для метода `get_context_data(self, **kwargs)` класса `DishesList` во `views` приложения `Dishes`.

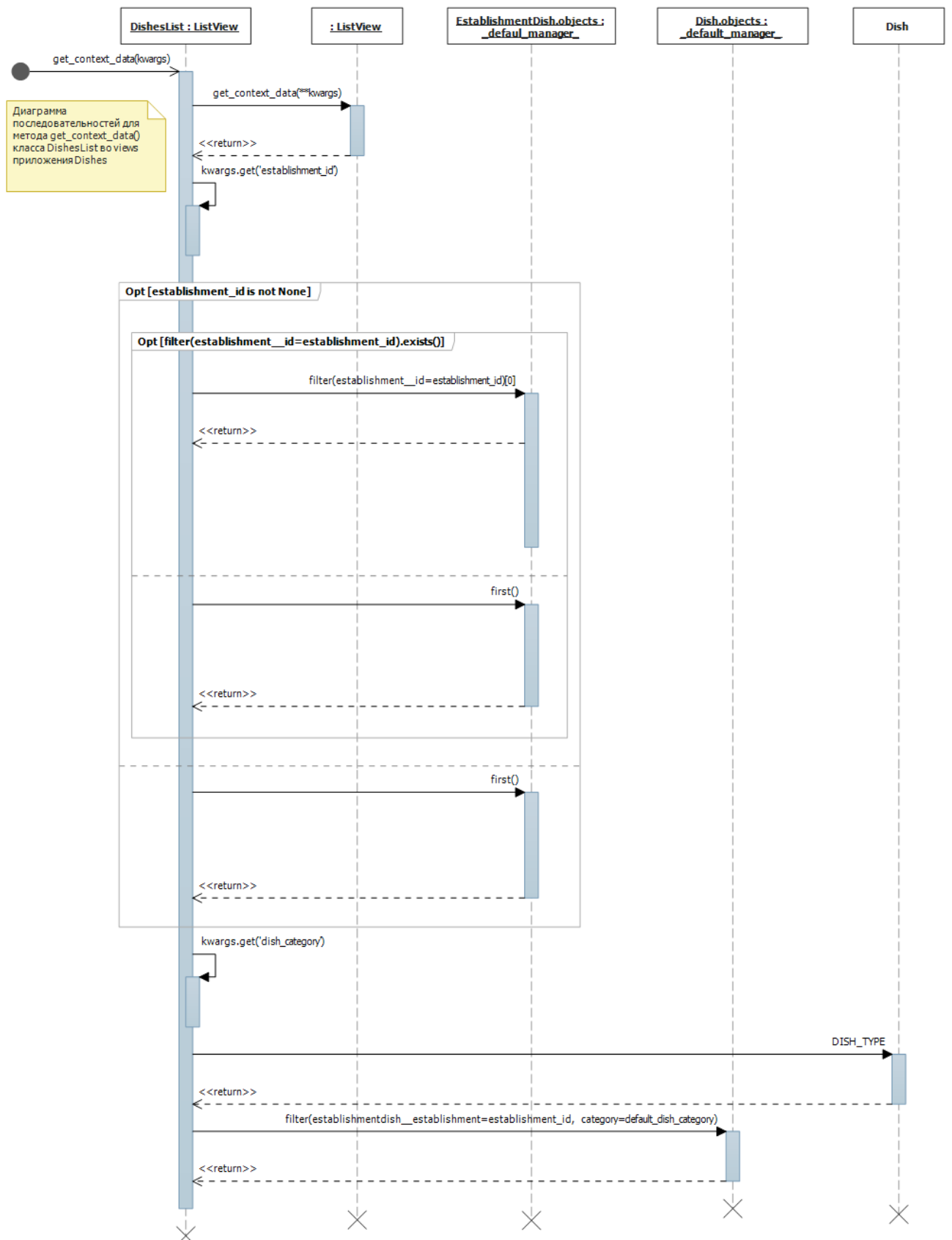


Рис. 13. Диаграмма последовательности `get_context_data(self, **kwargs)` в `DishesList`

Сценарий описываемой последовательности сообщений между объектами программной системы вызывается при вызове метода `get_context_data` файла «`views.py`» приложения «`dishes`». Вызов данной функции осуществляется при запросе по URL: `/establishment/<establishment_id>` или `/establishment/<establishment_id>/<dish_category>`, где переменные, обозначенные в скобках вида «`<...>`» являются параметрами запроса. При вызове метода `get_context_data(self, **kwargs)` создается объект `context`, которому передается результат вызова `super().get_context_data(**kwargs)`. После этого извлекается идентификатор заведения `establishment_id` с помощью функции `self.kwargs.get('dish_id')` из ассоциативного массива параметров запроса. Если данное значение успешно получено, то, при условии, что у объекта-списка `EstablishmentDish.objects` типа `_default_manager` имеется хотя бы один экземпляр объекта, соответствующего заведению с полученным идентификатором `establishment_id`, происходит извлечение этого объекта. В противном случае извлекается первый попавшийся объект `EstablishmentDish.objects`. Это де происходит, если не удалось извлечь идентификатор заведения `establishment_id`. После извлечения значения переменной текущей категории блюда из объекта `**kwargs`, происходит получение списка категорий блюд у объекта класса `Dish`. После чего происходит извлечение списка объектов-блюд определенного заведения и определенной категории у объекта-списка `Dish.objects` типа `_default_manager` с помощью функции `filter(establishmentdish__establishment=establishment_id, category=default_dish_category)`.

На Рис. 14 изображена диаграмма последовательности для метода `get_context_data(self, **kwargs)` класса `EmployeePage` во `views` приложения `Employees`.

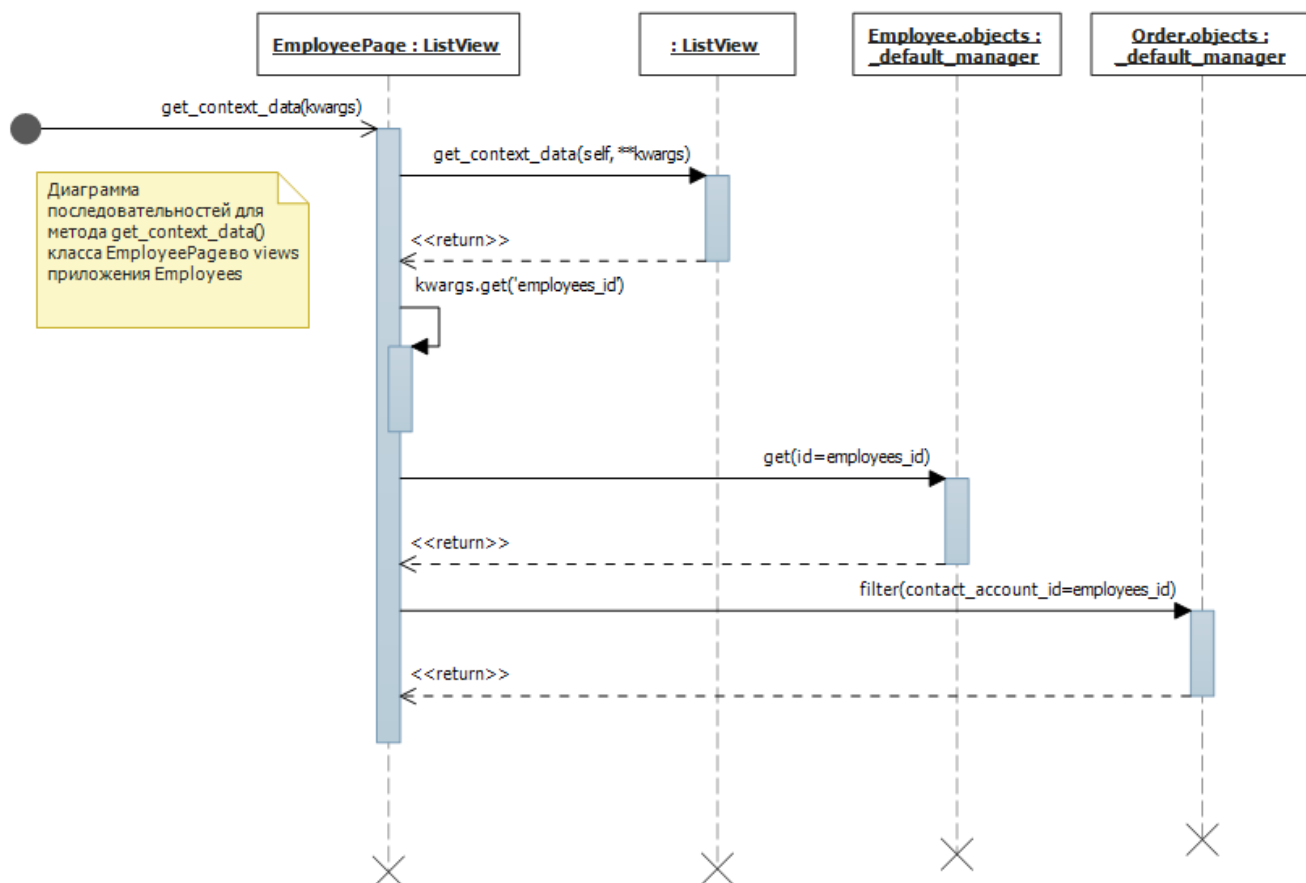


Рис. 14. Диаграмма последовательности `get_context_data(self, **kwargs)` в `Employees`

Сценарий описываемой последовательности сообщений между объектами программной системы вызывается при вызове метода `get_context_data` файла «`views.py`» приложения «`employees`». Вызов данной функции осуществляется при переходе приложения по следующему адресу: `url(r'(?P<employees_id>\d+)/$', EmployeePage.as_view(), name='employees')`. При вызове метода `get_context_data(self, **kwargs)` создается объект `context`, которому передается результат вызова `super().get_context_data(**kwargs)`. После этого извлекается идентификатор сотрудника `employees_id` с помощью функции `self.kwargs.get('employees_id')`. С помощью менеджера по умолчанию объектов `Employee` извлекается сотрудник заведения, соответствующий полученному идентификатору из строки запроса. В конце с помощью

менеджера по умолчанию объектов Order по идентификатору сотрудника создается объект order_list.

На Рис. 15 изображена диаграмма последовательности для метода login_view(request, login_state) во views приложения Employees.

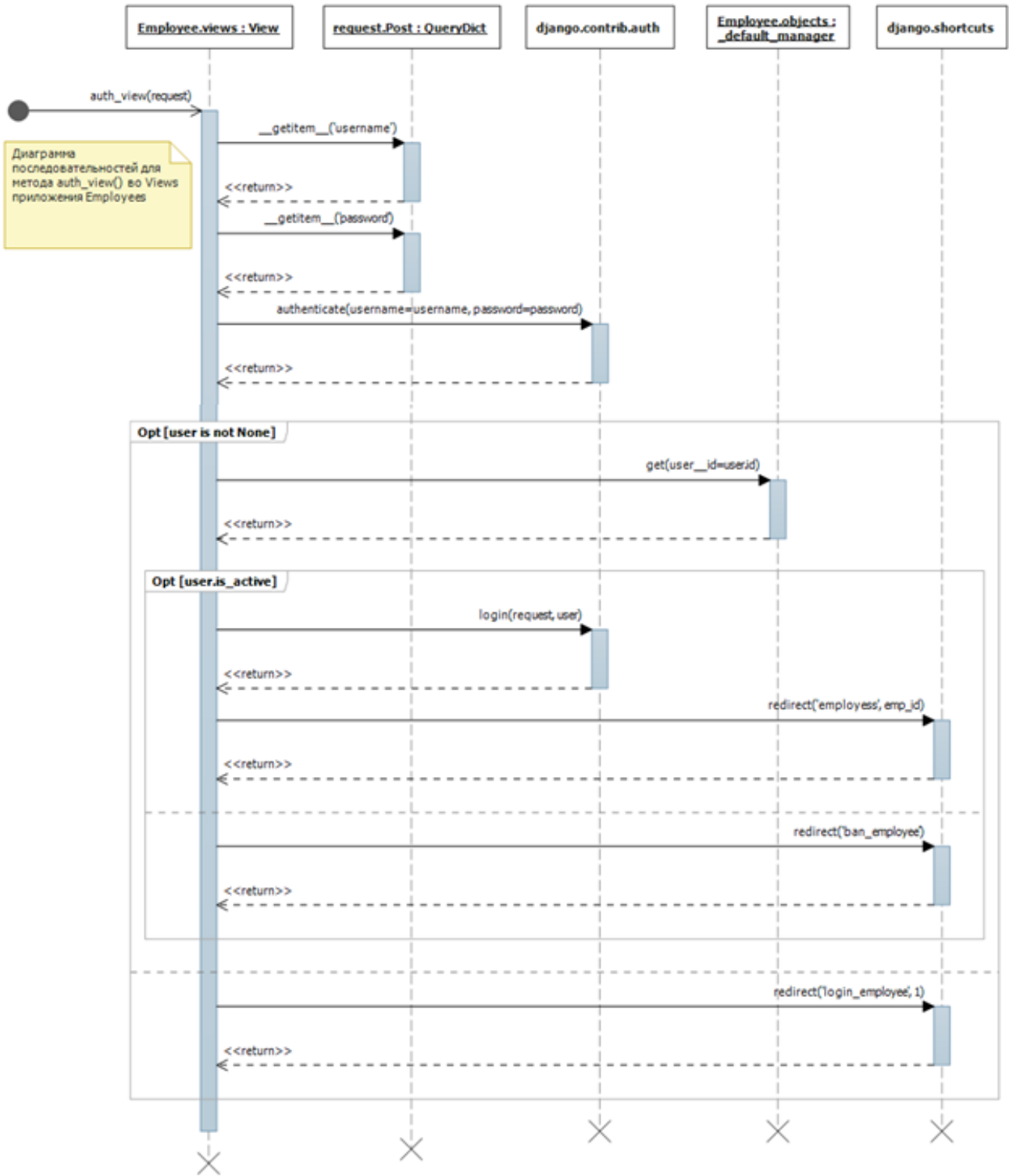


Рис. 15. Диаграмма последовательности login_view(request, login_state) в Employees

Сценарий описываемой последовательности сообщений между объектами программной системы вызывается при вызове метода `auth_view` файла «`views.py`» приложения «`employees`». Вызов данной функции осуществляется при переходе приложения по ссылке: `url(r'^accounts/after_login/$', views.auth_view, name='after_login')`, после нажатия кнопки «Войти» на странице `authentication.html`. После этого происходит получение значений ключей, переданных в POST запросе с помощью функции `__getitem__()` объекта `request.POST` типа `QueryDict`. Далее объекту `user` присваивается результат аутентификации по связке «логин-пароль» при помощи `auth.authenticate(username=username, password=password)`. После этого объект `user` проверяется по двум параметрам: во-первых – есть ли такой пользователь, а если нет, то происходит переход по ссылке `url(r'^accounts/login/(?P<login_state>\d+)?', views.login_view, name='login_employee')` на страницу `employees/authentication.html` и выводится предупреждение о неверном вводе имени или пароля. Если первая проверка прошла успешно, то пользователь проверяется на предмет наличия блокировки (`user.is_active`). Если пользователь не заблокирован, происходит его авторизация `auth.login(request, user)` и переход по ссылке `url(r'(?P<employees_id>\d+)/$', EmployeePage.as_view(), name='employees')` на страницу `employees.html`. Если пользователь не прошел проверку происходит переход по ссылке `url(r'^accounts/ban/$', views.ban_view, name='ban_employee')` на страницу `ban.html`.

На Рис. 16 - Рис. 17 изображена диаграмма последовательностей для метода `decrement_dish(request)` во `views` приложения `orders`.

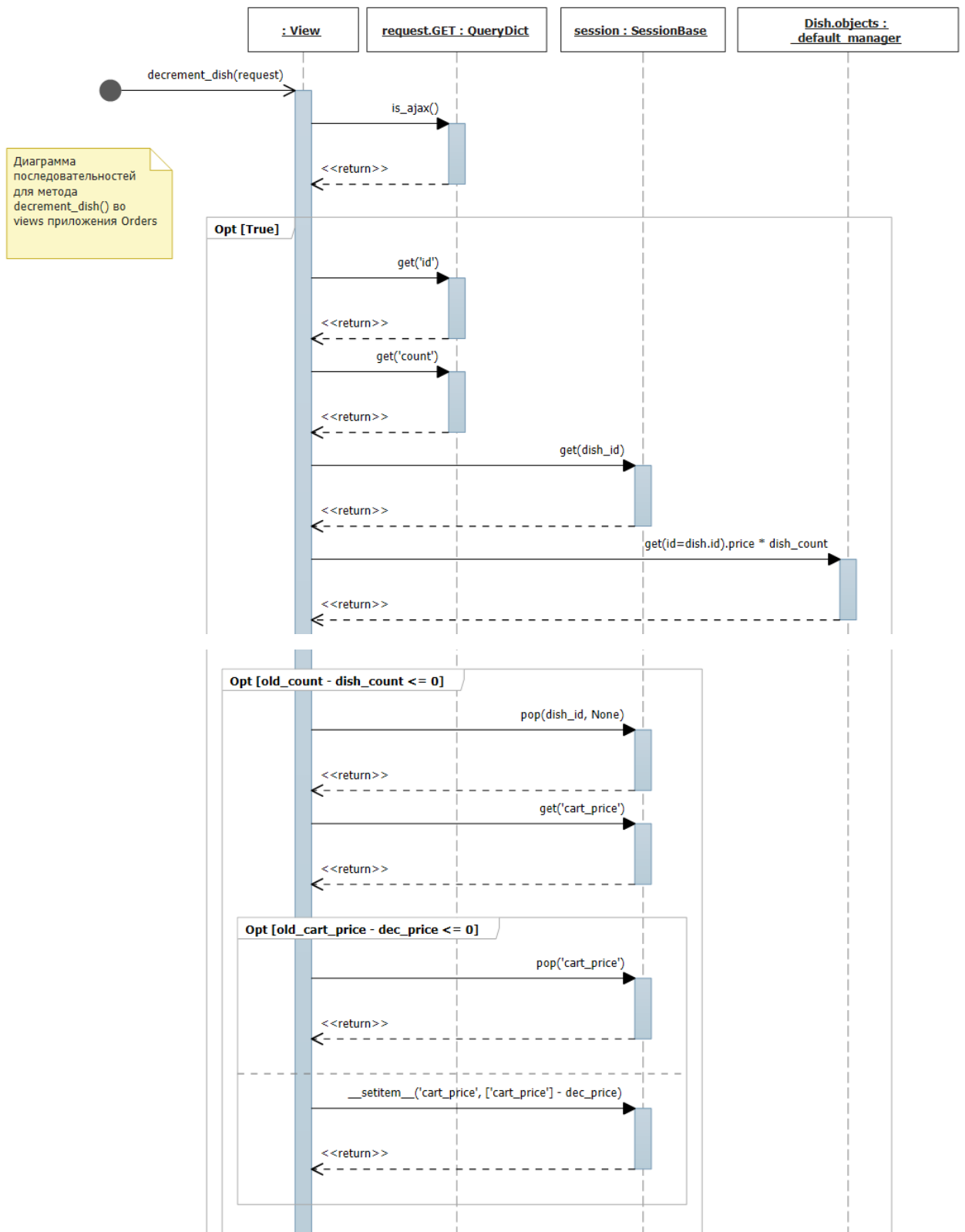


Рис. 16. Часть 1 диаграммы последовательностей для метода `decrement_dish(request)` в `orders`

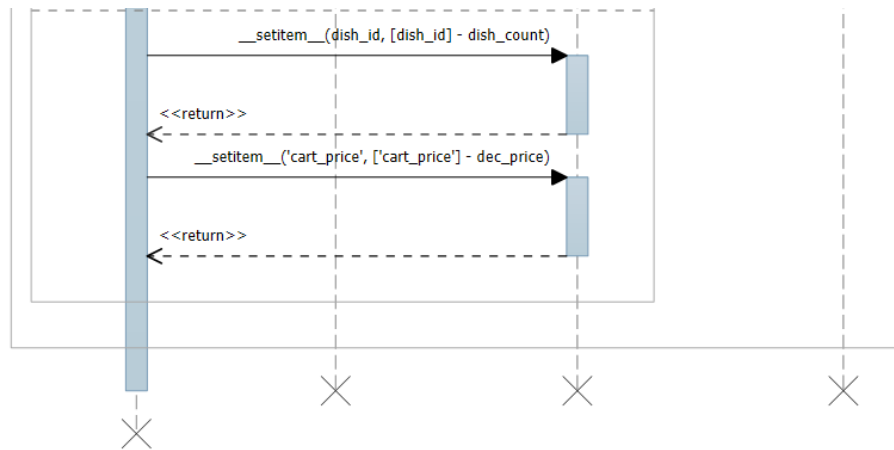


Рис. 17. Часть 2 диаграммы последовательностей для метода `decrement_dish(request)` в `orders`

Сценарий описываемой последовательности сообщений между объектами программной системы вызывается при вызове метода `decrement_dish` файла `views.py` приложения `orders`. Вызов данной функции осуществляется по запросу `ajax` со страницы просмотра корзины заказов, отсылаемому серверу по событию нажатия на кнопку уменьшения количества порций блюда.

Сценарий осуществляется только в том случае, если переданный запрос является запросом `ajax`. Это проверяется с помощью функции `is_ajax()` переданного в функцию объекта `request`. После этого происходит получение значений ключей, переданных в GET запросе с помощью функции `get()` объекта `request.GET` типа `QueryDict`. Также из объекта сессии (`session: SessionBase`) извлекается значение ключа – полученного из объекта `request` идентификатора блюда (`dish_id`) – которое является количеством порций данного блюда в заказе. С помощью объекта модели класса `Dish`, соответствующего полученному идентификатору, извлекается цена блюда. Если количество блюд в корзине стало равным нулю, то данный ключ удаляется из объекта сессии с помощью метода `pop(dish_id, None)`. В противном случае значение ключа, соответствующее количеству порций блюда, уменьшается на величину переданного в запросе значения количества с помощью метода `__setitem__`. Затем из объекта сессии извлекается цена

корзины заказов до нажатия на кнопку уменьшения порций блюд с помощью метода `get('cart_price')`. Если уменьшение цены корзины на величину цены выбранного блюда приводит к тому, что цена корзины становится равной нулю, то значение, содержащее в себе цену корзины, в объекте сессии удаляется с помощью метода `pop('cart_price', None)`. В противном случае значение данного ключа уменьшается на величину цены блюда с помощью метода `__setitem__` объекта сессии, устанавливающего новое значение для указанного ключа.

4 Реализация

4.1 Тестирование

Тестирование – это исследование на корректную работоспособность программы, через нахождение соответствия между реальными и ожидаемыми поведением программы, осуществляемые на наборе тестов. Также тестирование реализует отрицательную «обратную связь» при разработке. Она помогает доработать продукт, в противном случае отрицательная «обратная связь» в виде гневных отзывов конечных клиентов, не заставит себя долго ждать.

4.1.1 Модульные тесты

Вид тестирования, при котором проверяются модули программы. Идея разбить код программы на отдельные модули. Также модульное тестирование можно считать, как «живое» документирование. Это значит, что клиенты, которые не знают, как работает данный класс, могут разобраться с помощью модульного теста [8].

Файл test.py каждого приложения, который содержит тесты, создается в Django в ходе создания приложения в проекте. В данном файле можно создавать unit- и doc- тесты. В данной курсовой работе были выбраны unit-тесты для тестирования атрибутов и методов, которые их изменяют. Проще говоря, unit-тесты служат для непосредственной проверки заполненных полей, а не поэтапной работы программы.

Для создания unit-теста используется класс TestCase, от которого наследуются все созданные разработчиком классы с тестами.

1. Проверка свойства, задающего дату истечения заказа:

```
def test_order_expire_date(self):
    order_test = Order(
        client_phone=8432424,
        type=Order.TYPE_DINNER_WAGON,
        state=Order.STATE_DONE,
        order_date=date(2014, 12, 12),
        execute_date=datetime(2014, 12, 12).date()
    )
```

```

        self.assertEqual(
            order_test.expire_date,
            datetime(2015, 1, 11).date(),
            "expire_date is not equal"
        )

```

Данный тест проверяет работу свойства `expire_date` класса `Order`. Вызов данного свойства должен быть равен дате исполнения заказа, увеличенной на 30 дней.

2. Проверка поля, возвращающего название блюда:

```

def test_dish_class_name(self):
    dish = Dish(
        name='Гречка',
        price=70.50,
        category=Dish.DISH_TYPE_GARNISH,
    )
    self.assertEqual(
        dish.__str__(),
        'Гречка', "Имя класса неверно"
    )

```

В данном методе, удостоверяемся, что в ходе заполнения базы данных, поле `name` объекта `Dish` было передано верно.

3. Проверка метода, отменяющего заказ:

```

def test_order_decline(self):
    order_test = Order(
        client_phone=8432424,
        type=Order.TYPE_DINNER_WAGON,
        state=Order.STATE_DONE,
        order_date=date(2014, 12, 12),
        execute_date=datetime(2014, 12, 12).date(),
        dinner_wagon=DinnerWagon(seats=2)
    )
    order_test.decline()
    self.assertEqual(
        order_test.state,
        Order.STATE_CANCELED
    )

```

Данный тест проверяет поле `state` у заказа после его отмены, тем самым тестируя правильность работы метода.

4.1.2 Интеграционные тесты

Интеграционные тесты в Django создаются с помощью класса `WebTest`, который является наследником `TestCase`, который нужен для модульного

тестирования. Основное отличие `TestCase` и `WebTest` это переменная `self.testapp` из `DjangoTestApp`. Она позволяет получить доступ к API `WebTest`.

Важно понимать, что интеграционные тесты — это не замена юнит-тестам, а только дополнение к ним, и что 100% покрытие никак не гарантирует отсутствия ошибок. Юнит-тесты — точные, они говорят, что именно поломалось, они крайне полезны при рефакторинге и в сложных местах проекта.

Чтобы показать различие: в unit-тесте для формы регистрации мы бы создали объект для регистрации любого пользователя по e-mail, передавали бы в него разные словари с данными и смотрели бы, какие вызываются исключения. Юнит-тесты максимально приближены к листингу, тестируют отдельный его метод или атрибут, и позволяют проверить, что все части системы по отдельности работают правильно. Интеграционные тесты помогают проверять, что и вместе они работают тоже правильно [9].

4.1.3 Построение и выполнение тестов

Выполнение тестов осуществляется с помощью утилиты `manage.py test`. По умолчанию эта утилита проверит все тесты каждого приложения. Чтобы обратиться к какому-то определенному приложению и проверить только его тесты, следует использовать `python manage.py test <application_name>`.

Самое интересное в тестах Django версии 1.7 и выше – это то, что они не используют настоящую базу данных. Для тестов Django создает в оперативной памяти базу данных специально для тестов. И поля в ней заполняются теми данными, которые были использованы в тестах [10].

Запустим тесты во всем приложении:

```
C:\Kurovovoy>python manage.py test
Creating test database for alias 'default'...
.....
-----
-----
```

```
Ran 14 tests in 0.032s
```

```
OK
```

```
Destroying test database for alias 'default'..Затраченное  
время: 00:00:03.75
```

Из результатов запуска тестов видно, что все тесты, которых было запущено 14, прошли успешно.

4.1.4 Покрытие кода

Это инструмент для проверки покрытия кода тестами. Оно выражается в процентном соотношении тестируемых полей ко всем полям приложения. Покрытие кода используется для получения набора тестов при регрессивном тестировании, потому что в ходе тестирования выявляются недостатки, а также закрываются непокрытые участки кода.

В python для создания покрытия тестами используется пакет `coverage`. С её помощью можно получить отчет о покрытии в виде html-файла, xml-файла, вывести в консоль или текстовый файл.

На Рис. 18 представлен вывод команды `coverage` терминала PyCharm для данного проекта.

Name	Stmts	Miss	Cover
dishes__init__	0	0	100%
dishes\admin	4	3	25%
dishes\models	28	2	93%
dishes\tests	14	0	100%
employees__init__	0	0	100%
employees\admin	3	2	33%
employees\models	18	9	50%
employees\tests	14	13	7%
establishments__init__	0	0	100%
establishments\admin	7	0	100%
establishments\models	35	9	74%
establishments\tests	17	13	24%
manage	6	0	100%
orders__init__	0	0	100%
orders\admin	4	3	25%
orders\models	56	21	62%
orders\tests	26	25	4%
wanna_eat__init__	0	0	100%
wanna_eat\settings	25	14	44%
TOTAL	278	114	59%

Рис. 18. Вывод результата покрытия кода тестами

Некоторые тестировщики считают, что делать 100% покрытие кода unit-тестами не нужно, так как это бесполезно и тратит время. Но, чтобы добиться защищенного и качественно работающего приложения, потому что приложения в современном мире постоянно обновляются, соответственно, после каждого обновления поля, которые были созданы в более старых версиях, могут работать некорректно. А при полном покрытии кода все возможные ошибки будут найдены.

4.1.5 Запуск приложения для тестирования

Запуск приложения осуществляется на сервере разработки. Этот сервер служит непосредственно для разработки приложения. Ему не нужно много ресурсов, в отличие от сервера для непосредственной работы приложения Apache Tomcat, например.

Сервер запускается на 127.0.0.1:8000. Это может вызвать небольшие проблемы, так как, например, Skype, приложению для видео коммуникации,

работает именно на 8000 порту. В целях разрешения возможных конфликтов следует перенести сервер, например, на 127.0.0.1:8080. Для этого используем команду `python manage runserver 8080`. Приложение будет расположено именно на этом порту.

Запуск и отладка проекта на локальном тестовом сервере для разработки необходим для выявления правильности работы приложения, тестирования пользовательского интерфейса и отладки кода программы.

4.2 Непрерывная интеграция

В разработке программного обеспечения непрерывная интеграция – это практика частой сборки и тестирования проекта с целью выявления ошибок на ранней стадии интеграции проекта. Непрерывная интеграция — автоматизированный процесс, в котором, как правило, используется специализированное серверное ПО, отвечающее за поиск изменений в коде в системе контроля версий, сборку, развертывание и тестирование приложения. Эти действия система непрерывной интеграции совершает автоматически при каждом обновлении репозитория системы контроля версий [11].

Поскольку в качестве системы контроля версий для данного курсового проекта выбрана система Git с расположением репозитория проекта на бесплатном (для публичных репозиторий) хостинге GitHub, то в качестве системы непрерывной интеграции выбран удобный для таких задач сервис Travis CI. Travis CI – это распределенный веб сервис для сборки и тестирования проектов по разработке программного обеспечения на более чем 20-ти языках программирования, использующих GitHub в качестве хостинга исходного кода [12].

Процесс настройки системы непрерывной интеграции и первая сборка проекта представляет собой простую последовательность действий, описанную ниже. Для того, чтобы осуществить привязку своего репозитория на GitHub необходимо быть хозяином этого репозитория.

Первым шагом является вход в учетную запись Travis CI через аккаунт на GitHub. Для этого необходимо с сайта сервиса (<https://travis-ci.org/>) нажать на кнопку, представленную на Рис. 19.

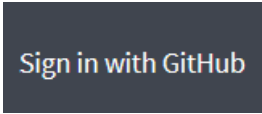
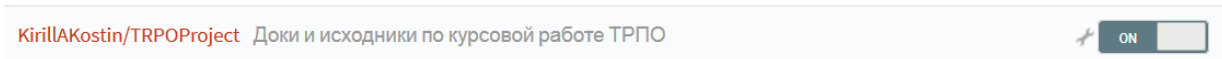


Рис. 19. Кнопка входа в учетную запись Travis CI

Следующим шагом является активация синхронизации репозитория проекта на GitHub с сервисом Travis CI. Для этого необходимо зайти в настройки репозитория профиля на Travis CI и включить работу сервиса с выбранным репозиторием с помощью переключателя, изображенного на Рис. 20.



KirillAKostin/TRPOProject Доки и исходники по курсовой работе ТРПО

ON

Рис. 20. Переключатель Travis CI для работы с выбранным репозиторием

Следующим шагом, указывающим сервису Travis то, как работать с репозиторием, является добавление в корневую директорию репозитория файла `.travis.yml`. Данный файл содержит скрипт, указывающий сервису то, как настроен репозиторий проекта: на каком языке написан исходный код, что нужно выполнить перед сборкой проекта, какие скрипты нужно выполнить после сборки проекта и другие опции, о которых можно узнать на сайте Travis CI [13].

Содержание файла `.travis.yml` для данного курсового проекта представлено ниже.

```
language: python
python:
- "3.4"
install: "pip install -r requirements.txt"
script:
- "flake8"
- "coverage run manage.py test"
- "coverage report"
notifications:
email:
```

```
|         on_failure:  
|         - kiraarik@sibmail.com
```

Как видно из содержания файла, в нем указывается, что проект написан на языке python. Это необходимо для того, чтобы Travis смог правильно интерпретировать проект и создать необходимое виртуальное окружение на сервере для сборки проекта. Далее указывается версия интерпретатора python, необходимая для выполнения установок и скриптов на сервере.

После этого указываются секции, которые выполняет Travis CI в проекте. В секции `install` он выполняет скрипт по установке необходимых зависимостей для корректной работы программной системы. В секции `scripts` указываются скрипты, которые также необходимо выполнить после сборки проекта и установки всех зависимостей. В данном случае выполняется скрипт `flake8`, проверяющий стиль написания кода на python (настройки для выполнения данного скрипта находятся в файле проекта `tox.ini`), затем запускаются тесты и ведется подсчет процента покрытия кода тестами с помощью команд `coverage`.

При возникновении ошибок сборки Travis CI высылает на почту пользователя, который сделал коммит, на котором возникла ошибка, уведомление о том, что его коммит привел к нарушению работы проекта. В файле Travis CI данного проекта также указывается e-mail, на который, в случае возникновения ошибок, всегда будет высылаться уведомление.

После настройки файла `.travis.yml` для проверки того, работает ли Travis CI с репозиторием необходимо зайти в настройки репозитория, во вкладку «Webhooks and Services» и выбрать сервис Travis CI, на открывшейся странице нужно нажать на кнопку «Test service» (Рис. 21), чтобы убедиться в работоспособности сервиса с вашим репозиторием.

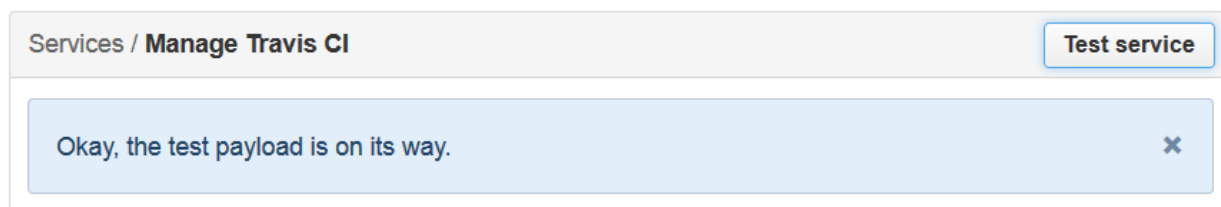


Рис. 21. Тестирование сервиса Travis CI на GitHub

Сама по себе такая проверка не инициирует сборку проекта. Для запуска построения проекта необходимо сделать коммит и отправить его в репозиторий на сервер GitHub.

Увидеть то, как прошла сборка и логи сборки можно на сайте Travis CI, зайдя под своей учетной записью. Если все команды завершились с кодом 0, то сборка проекта считается выполненной успешно и её присваивается статус «Passed», в противном случае ей присваивается статус «Failed».

5 Документация

5.1 Назначение программы

Разработанный проект «Wanna_Eat» предназначен для оформления заказов различных заведений нескольких городов. При этом клиенту для совершения покупок нет необходимости регистрироваться в системе, достаточно указать свой номер телефона при оформлении заказа.

В частности данный сайт позволяет:

- Просматривать меню различных заведений разных городов;
- Добавлять блюда в корзину, изменять количество порций, удалять выбранное блюдо из корзины;
- Оформлять заказы на выбранный список блюд;
- Выбирать типы заказов: заказ столика, заказ доставки или самовывоз;
- Отслеживать состояние выполнения заказов, используя номер телефона, указанный при оформлении заказа;
- Работникам заведений просматривать и обрабатывать списки заказов.

Сайт «Wanna_Eat» рассчитан на людей, имеющих доступ в интернет и желающих получать детальную информацию о меню различных заведений и осуществлять заказы, не выходя из дома.

5.2 Условия запуска программы

Для работы на сайте пользователю необходимо иметь персональный компьютер, смартфон или любое другое техническое устройство с возможностью выхода в интернет со скоростью не менее 256 КБ/с. Также необходим установленный браузер и возможность ввода.

В ходе самой эксплуатации есть ряд ограничений, такие как:

- При оформлении заказа обязательными полями являются: «Дата», «Время», «Контактный телефон»;

- При заказе доставки обязательным полем является «Адрес доставки»;
- При оформлении заказа нельзя указывать дату и время, позже, чем текущее время + 2 часа – это время, необходимое для обработки заказа сотрудником заведения;
- При оформлении заказа номер телефона не может состоять из букв, а длина номера должна быть равна 10 цифрам.

5.3 Выполнение программы

Сервис выполняет следующие функции:

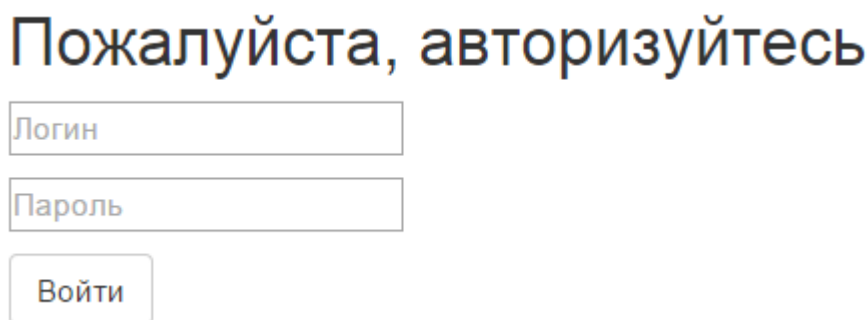
5.3.1 Авторизация работника заведения

Для прохождения авторизации необходимо кликнуть по кнопке «Войти» в правом нижнем углу любой страницы (Рис. 1).

Войти

Рис. 22. Кнопка «Войти»

Затем открывается новая страница с полями: Логин и Пароль. Эта страница показана на Рис. 23.



Пожалуйста, авторизуйтесь

Логин

Пароль

Войти

Рис. 23. Страница авторизации для работника заведения

Каждое поле является обязательным. После заполнения каждого поля, Работник заведения нажимает кнопку «Войти», после чего он получает доступ к другим функциям сайта.

5.3.2 Просмотр списка заказов работником заведения

После авторизации работник заведения имеет доступ к странице заказов этого заведения. Здесь же заказ можно принять или отменить (Рис. 24).

Номер заказа: 24	Принять
Тип заказа: Бронирование столика	Отменить
Статус заказа: Выполняется	Подробнее
Контактный номер клиента: 2222221111	
Дата исполнения заказа: Dec. 31, 2014 3 p.m.	

Рис. 24. Страница списка заказов заведения

5.3.3 Выбор блюд по категории

Все блюда меню относятся к определенной категории блюд, список категорий представлен на Рис. 25.

Все
Алкогольные напитки
Безалкогольные напитки
Гарниры
Горячие блюда
Дессерты
Закуски
Салаты
Пиццы
Роллы
Супы

Рис. 25. Список категорий блюд

Клиент может выбрать категорию (по умолчанию «Все»). Список блюд заведения фильтруется согласно выбранной категории.

5.3.4 Добавление блюда в корзину

Для добавления блюда в корзину необходимо кликнуть по кнопке «Добавить в корзину» (Рис. 26).

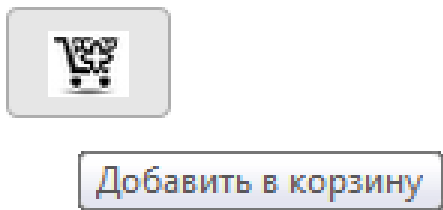


Рис. 26. Кнопка «Добавить в корзину»

При этом одна порция выбранного блюда будет добавлена в корзину. При повторном выборе блюда в корзину будет добавляться одна порция за клик.

5.3.5 Работа с корзиной

Для того чтобы перейти в корзину заказов, необходимо кликнуть по иконке корзина, или ссылке с соответствующей надписью (Рис. 27).

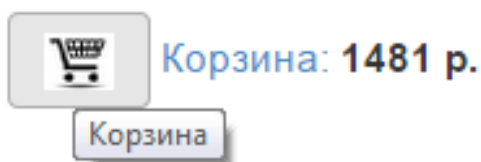
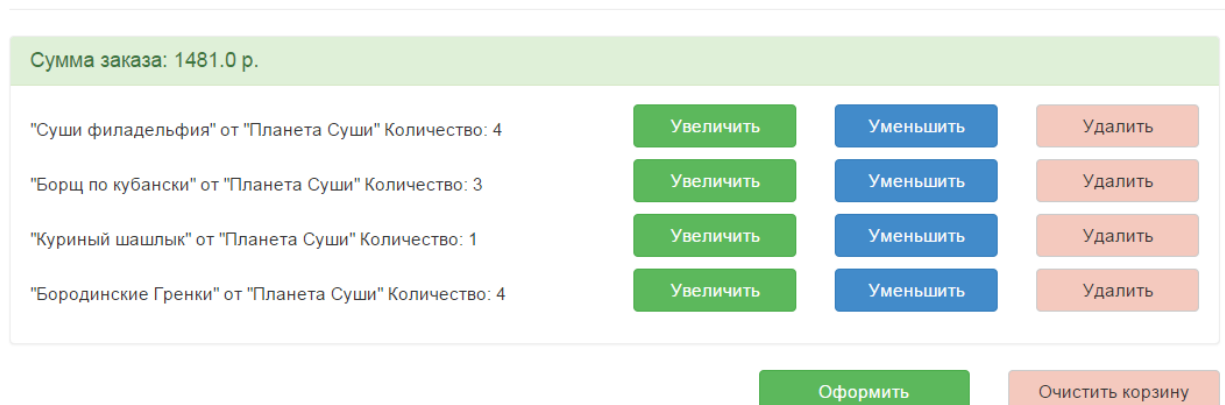


Рис. 27. Кнопка и текстовая ссылка «Корзина»

Сама корзина содержит в себе список блюд с указанием заведения, в котором было выбрано блюдо и количества порций (Рис. 28).

Корзина:



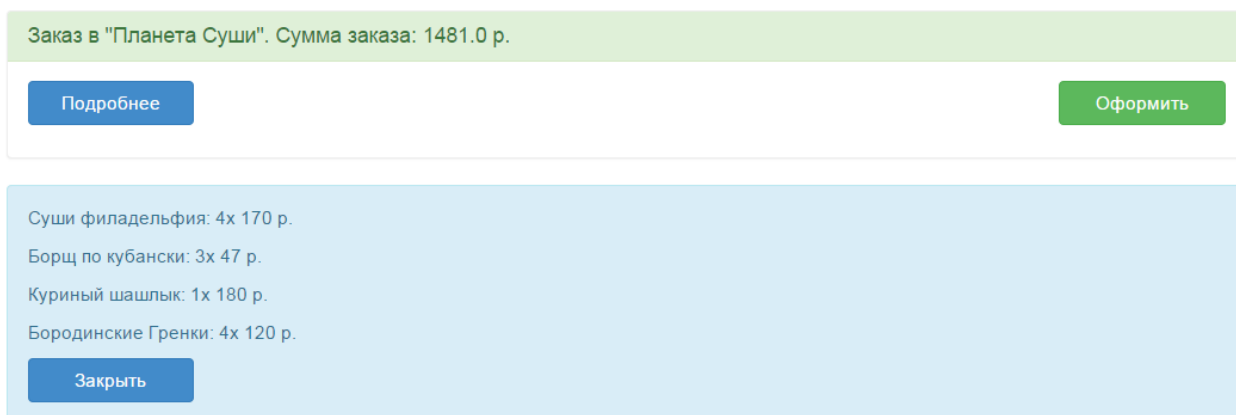
Сумма заказа: 1481.0 р.			
"Суши филадельфия" от "Планета Суши" Количество: 4	Увеличить	Уменьшить	Удалить
"Борщ по кубански" от "Планета Суши" Количество: 3	Увеличить	Уменьшить	Удалить
"Куриный шашлык" от "Планета Суши" Количество: 1	Увеличить	Уменьшить	Удалить
"Бородинские Гренки" от "Планета Суши" Количество: 4	Увеличить	Уменьшить	Удалить
Оформить		Очистить корзину	

Рис. 28. Корзина заказов

В корзине клиенту предоставляется возможность редактирования количества порций блюда или же исключения одного из меню вовсе. Также возможна полная очистка корзины. Когда выбор сделан, клиенту необходимо кликнуть по кнопке «Оформить».

5.3.6 Формирование заказов

После переход по ссылке «Оформить» корзины заказов система на основе данных корзины формирует один или несколько заказов для клиента (в зависимости от количества заведений, в которых был сделан выбор). Клиент может посмотреть детальную информацию по каждому заказу (Рис. 29, кнопка «Подробнее»).



Заказ в "Планета Суши". Сумма заказа: 1481.0 р.

Подробнее

Оформить

Суши филадельфия: 4x 170 р.
Борщ по кубански: 3x 47 р.
Куриный шашлык: 1x 180 р.
Бородинские Гренки: 4x 120 р.

Закрыть

Рис. 29. Детальная информация о заказе

Здесь же клиент может оформить каждый заказ, кликнув по кнопке «Оформить».

5.3.7 Оформление заказа

После перехода по ссылке оформления заказа клиент переадресовывается на страницу «Оформление заказа». Здесь необходимо выбрать тип заказа: «Бронирование столика», «Заказ самовывоза» или «Заказ доставки» (по умолчанию – «Бронирование столика»). Если клиент выбрал «Бронирование столика», то ему необходимо заполнить ряд полей формы (Рис. 30). Часть полей имеют значения по умолчанию, но дату бронирования, время и контактный телефон пользователю необходимо заполнить в любом случае.

The screenshot shows a web form titled 'Детали заказа' (Order Details). On the right, there is a blue button 'Бронирование столика' (Table Reservation) which is selected, and two other buttons: 'Заказ самовывоза' (Self-pickup order) and 'Заказ доставки' (Delivery order). The main form area contains the following fields and controls:

- Адрес заведения** (Restaurant address): A dropdown menu showing 'ул. Главная 154' (Main St. 154) with an asterisk indicating it is required.
- Тип зала заведения** (Restaurant hall type): A dropdown menu showing 'Не курящий' (Non-smoking) with an asterisk.
- Дата бронирования** (Reservation date): A date input field showing '28.12.2014' with an asterisk.
- Показать столики** (Show tables): A button.
- Время бронирования** (Reservation time): A time input field showing '18:00' with an asterisk.
- Количество мест за столиком** (Number of seats at the table): A dropdown menu showing '4' with an asterisk.
- Если поле пустое, то свободных столиков нет** (If the field is empty, there are no free tables): A text label.
- Ваш контактный телефон** (Your contact phone): An input field showing '222221111' with an asterisk.
- Необходим для обработки и подтверждения заказа сотрудниками заведения.** (Required for processing and confirmation of the order by the establishment staff.): A text label.
- Оформить заказ** (Place order): A button.

Рис. 30. Форма заказа «Бронирование столика»

Если клиент выбрал «Заказ самовывоза» (Рис. 31), то ему также необходимо выбрать дату самовывоза, время и контактный телефон.

Детали заказа

Адрес заведения

просп. Комсомольский 153 ▼ *

Дата самовывоза

25.12.2014 *

Время самовывоза

18:00 *

Ваш контактный телефон

2222221111 *

Необходим для обработки и подтверждения заказа сотрудниками заведения.

Оформить заказ

Бронирование столика

Заказ самовывоза

Заказ доставки

Рис. 31. Форма заказа «Заказ самовывоза»

Если клиент выбрал «Заказ доставки» (Рис. 32), то ему также необходимо выбрать дату самовывоза, время и контактный телефон, а также адрес доставки.

Детали заказа

Дата доставки

31.12.2014 *

Время доставки

18:00 *

Адрес для доставки

ул. Вершинина, 39а *

Адрес в пределах выбранного города.

Ваш контактный телефон

2222221111 *

Необходим для обработки и подтверждения заказа сотрудниками заведения.

Оформить заказ

Бронирование столика

Заказ самовывоза

Заказ доставки

Рис. 32. Форма заказа «Заказ доставки»

Во всех случаях клиент может указать дату двумя способами – ввести вручную или использовать виджет «Календарь» (Рис. 33).

Дата бронирования

28.12.2014 x ▼ *

Декабрь 2014 ▼

◀

●

▶

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Рис. 33. Виджет «Календарь»

Если оформление заказа прошло успешно, пользователь получит уведомление (Рис. 34).

Ваш заказ успешно оформлен.

Заказ поступил на рассмотрение администрацией заведения. Они могут связаться с Вами по указанному в заказе телефону.

Вы можете посмотреть свои заказы в меню "Мои заказы" нашего сайта.

Вы также можете сами связаться с администрацией заведения:

Электронная почта: sushiplanet@rest.tomsk.ru

Телефон заказов: 9878987897

Телефон для справок: 9999999999

[Вернуться в меню](#)[Сделать еще заказ](#)

Рис. 34. Сообщение об успешном оформлении заказа

5.3.8 Просмотр заказов клиентом

Клиент, используя номер телефона, указанный в заказе как контактный, может просмотреть статус своих заказов (Рис. 35).

Получить заказы по номеру телефона

Ваш контактный телефон
 *

Бронирование столика			
Статус: Выполняется	Планета Суши - sushiplanet@rest.tomsk.ru ул. Главная 154 - 9999999999	Заказ	Dec. 31, 2014 3 p.m.

Бронирование столика			
Статус: На рассмотрении	Планета Суши - sushiplanet@rest.tomsk.ru ул. Главная 154 - 9999999999	Заказ	Dec. 28, 2014 3 p.m.

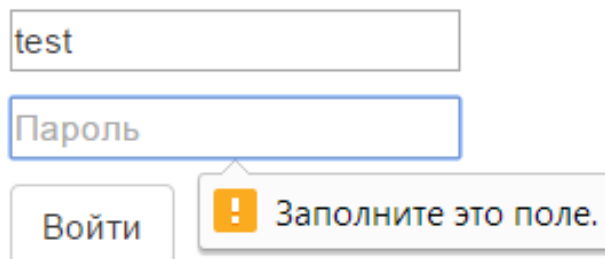
Рис. 35. Просмотр заказов клиентом

5.4 Сообщения программы

При выполнении различных операций на сайте, могут возникать сообщения обозначающие неверные действия пользователя. Некоторые поля являются обязательными или на них стоят иные ограничения, и, при не

заполнении этих полей или вводе некорректных данных, пользователь получает соответствующее уведомление.

Так, например, если поля Логин или Пароль пусты, а работник заведения пытается авторизоваться, то система выдаст предупреждение (Рис. 36).



The image shows a login interface. At the top, a text input field contains the word "test". Below it, a password input field is labeled "Пароль" and is currently empty. To the left of the password field is a button labeled "Войти". To the right of the password field, a yellow warning icon (an exclamation mark inside a square) is displayed next to the text "Заполните это поле." (Fill in this field.).

Рис. 36. Предупреждение о пустом поле

Если комбинация Логин-Пароль была указана неверно, на странице появится оповещение об этом (Рис. 37).

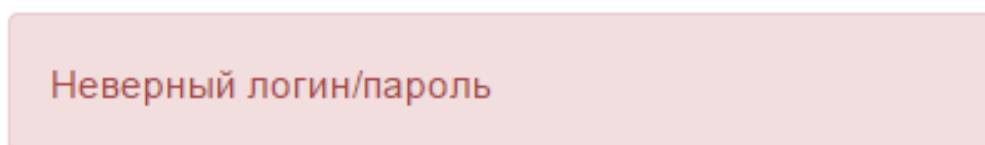


Рис. 37. Оповещение о неверной комбинации Логин-Пароль

Если комбинация была указана верно, но аккаунт по каким-либо причинам был заблокирован, работник заведения получит уведомление об этом (Рис. 38).

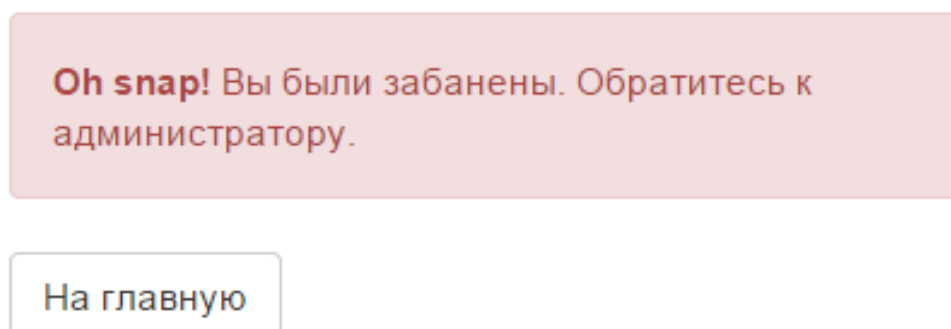


Рис. 38. Оповещение о блокировке аккаунта

Если при оформлении заказа одно или несколько обязательных полей были не заполнены, клиент получит уведомление об этом напротив некорректно заполненного поля (Рис. 39).

Ваш контактный телефон

9004001020

* Поле номера телефона не может быть пустым

Рис. 39. Пример сообщения о некорректном вводе данных на формах оформления заказов

Заключение

В данной курсовой работе был спроектирован и разработан прототип программного продукта для оформления заказов в ресторанах, кафе, барах и других заведениях нескольких городов. Проект был разработан с использованием языка программирования Python и фреймворка для создания веб приложений на языке python Django. Для контроля версий продукта и облегчения работы разработчиков в команде при разработке проекта была использована система контроля версий Git и хостинг GitHub.

Несмотря на то, что проект является небольшим, а на разработку было потрачено немного времени, большое внимание было уделено анализу и проектированию системы, что характерно для использования тяжеловесных методологий разработки и крупных проектов промышленного масштаба. Однако, развитие способностей анализа системы, проектирования компонентов и исследования предметной области является незаменимым для любых типов разработки программного обеспечения и повышает не только компетенции разработчика, но и качество его работы, навыки работы в команде и, как следствие, качество конечных продуктов.

Разработанный программный продукт является лишь прототипом, который можно улучшать, подстраиваясь под требования и пожелания возможных заказчиков и конкретных потребителей. Например, возможен выпуск версий приложения для мобильных систем, используя возможности Django, который без изменений в исходном коде программы позволит создать API для более удобной работы с сервисом с помощью любого устройства.

Список использованных источников

1. Ресторанный бизнес [Электронный ресурс] / openbusiness.ru – ведущий российский портал бизнес-планов, руководств и франшиз – Электрон. Дан. – URL: <http://www.openbusiness.ru/html/restoran1.htm>, свободный (дата обращения 10.12.2014).
2. Почему Django [Электронный ресурс] / OLDMIN TEAM – создание, поддержка и обслуживание сайтов – Электрон. Дан. – URL: <http://oldmin.org/ru/article/pochemu-django/>, свободный (дата обращения 10.12.2014).
3. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 624 с., ил.
4. Макконнелл С. Совершенный код – Пер. с англ. – СПб: Питер, 2007. – 896 с.
5. Диаграмма пакетов UML [Электронный ресурс] / Планерка – креативные решения в проектировании, бизнесе и в частной жизни – Электрон. Дан. – URL: <http://www.planerka.info/item/Diagrammy-paketov>, свободный (дата обращения 10.12.2014).
6. Диаграмма последовательности [Электронный ресурс] / Википедия – свободная энциклопедия – Электрон. Дан. – URL: http://ru.wikipedia.org/wiki/Диаграмма_последовательности, свободный (дата обращения 10.12.2014).
7. Теория и практика UML. Диаграмма последовательности [Электронный ресурс] / IT-GOST.RU – электронная библиотека стандартов оформления проектной документации – Электрон. Дан. – URL: http://it-gost.ru/articles/view_articles/94, свободный (дата обращения 10.12.2014).
8. Модульное тестирование [Электронный ресурс] / Википедия – свободная энциклопедия – Электрон. Дан. – URL: http://ru.wikipedia.org/wiki/Модульное_тестирование, свободный (дата обращения 10.12.2014).

9. Пишем функциональные/интеграционные тесты для проекта на django [Электронный ресурс] / Хабрахабр – самое крупное в Рунете сообщество людей, занятых в индустрии высоких технологий – Электрон. Дан. – URL: <http://habrahabr.ru/post/91471/>, свободный (дата обращения 10.12.2014).

10. Пишем функциональные/интеграционные тесты для проекта на django [Электронный ресурс] / Документация Django 1.4 (читать 1.7) – Электрон. Дан. – URL: <http://djbook.ru/rel1.4/topics/testing.html#the-test-database>, свободный (дата обращения 10.12.2014).

11. Непрерывная интеграция [Электронный ресурс] / CustIS Заказные ИнформСистемы – Электрон. Дан. – URL: http://lib.custis.ru/Непрерывная_интеграция, свободный (дата обращения 10.12.2014).

12. Travis CI [Электронный ресурс] / Википедия – свободная энциклопедия – Электрон. дан. – URL: https://ru.wikipedia.org/wiki/Travis_CI/, свободный (дата обращения: 10.12.2014).

13. Building a Python Project [Электронный ресурс] / Travis CI: Travis CI Documentation – Электрон. дан. – URL: <http://docs.travis-ci.com/user/languages/python/>, свободный (дата обращения: 10.12.2014).

Приложение А

Расчет КТУ

Заполнила С.А. Агеева

Таблица 1

	Агеева	Гаврилов	Костин	Рафиков
Агеева	1	1	0	1
Гаврилов	1	1	0	1
Костин	2	2	1	2
Рафиков	1	1	0	1

Заполнил К.А. Гаврилов

Таблица 2

	Агеева	Гаврилов	Костин	Рафиков
Агеева	1	0	0	1
Гаврилов	2	1	0	2
Костин	2	2	1	2
Рафиков	1	0	0	1

Заполнил К.А. Костин

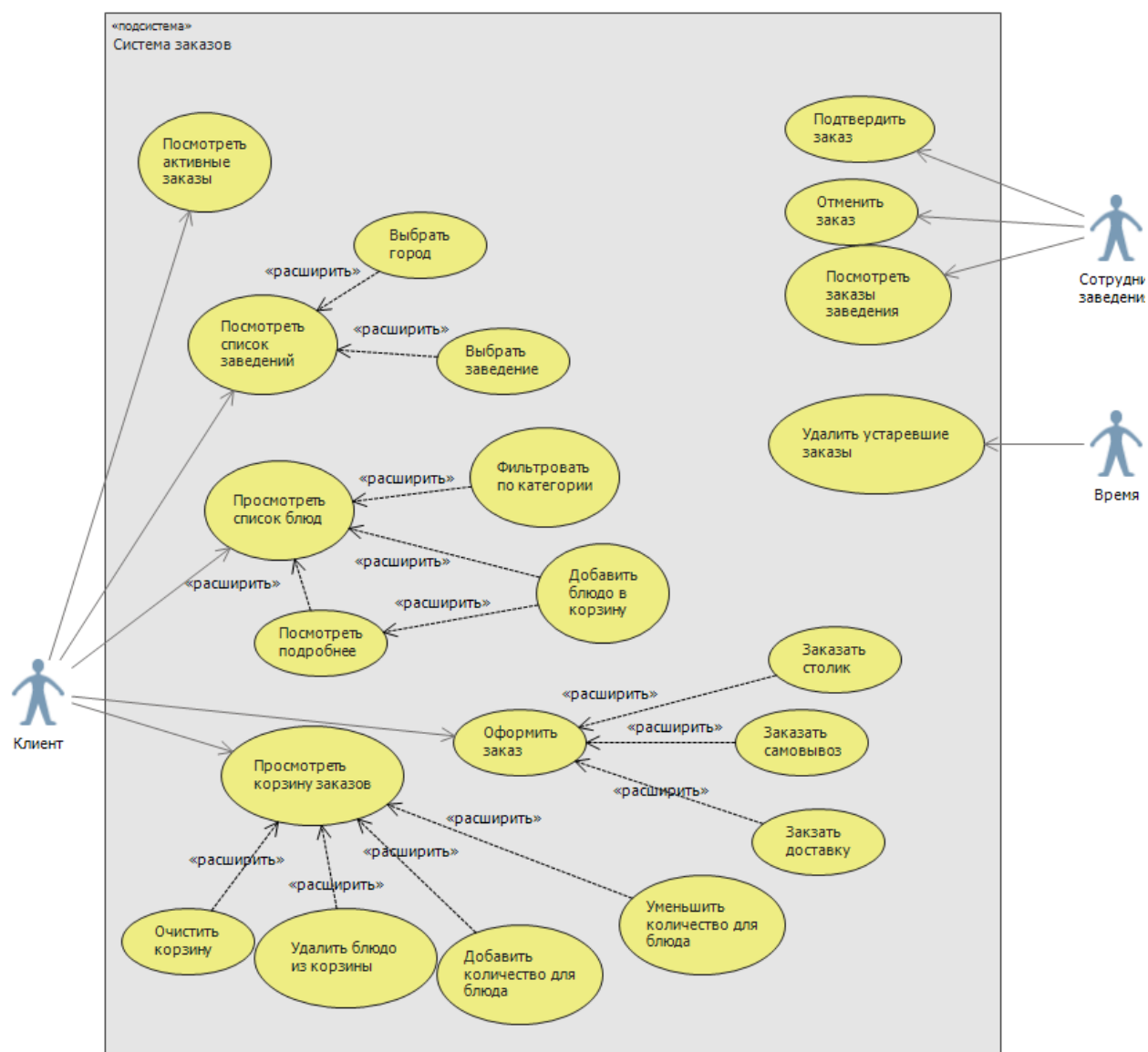
Таблица 3

	Агеева	Гаврилов	Костин	Рафиков
Агеева	1	0	0	1
Гаврилов	2	1	0	2
Костин	2	2	1	2
Рафиков	1	0	0	1

	Агеева	Гаврилов	Костин	Рафиков
Агеева	1	1	0	1
Гаврилов	1	1	0	1
Костин	2	2	1	2
Рафиков	1	1	0	1

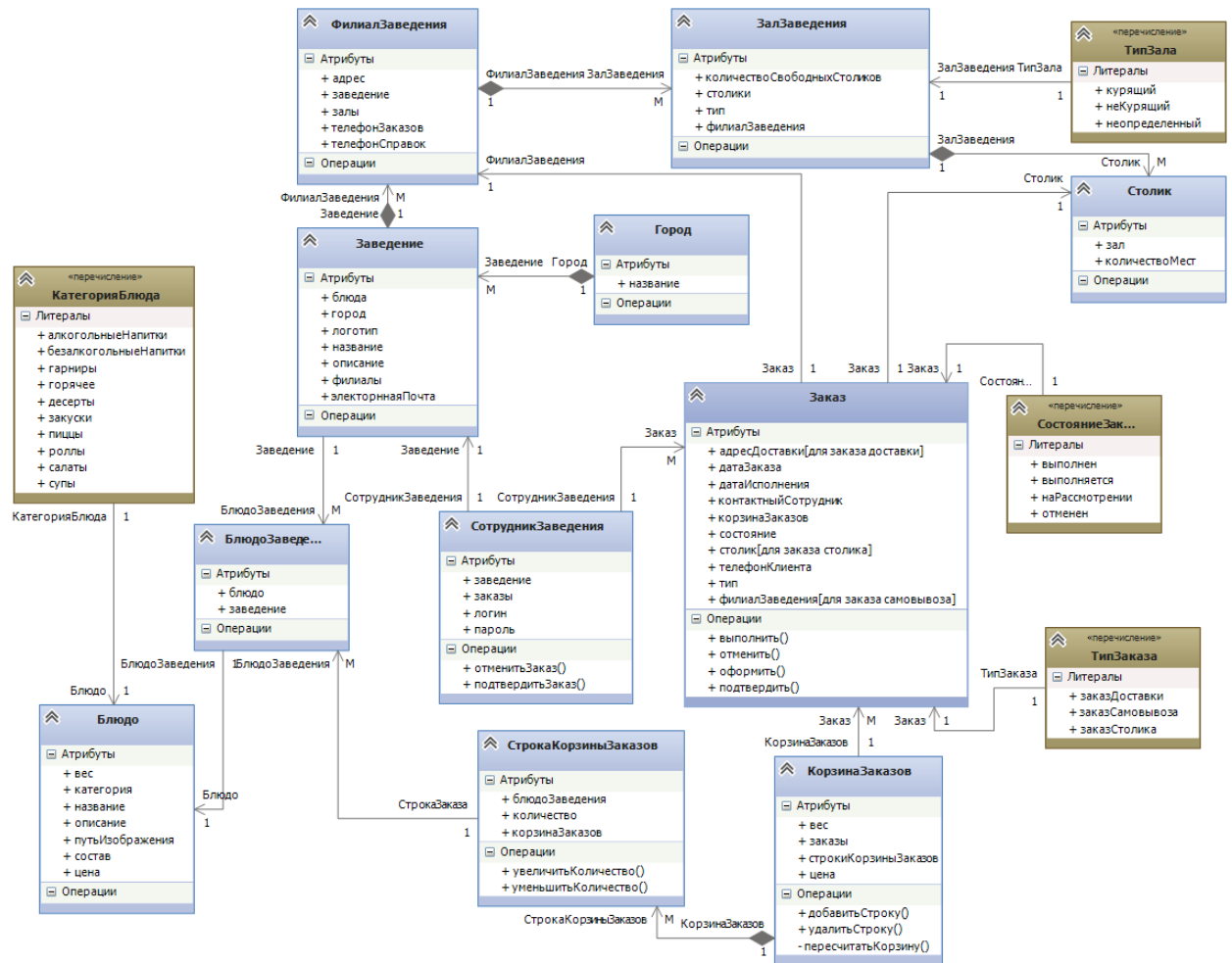
Приложение Б

Диаграмма вариантов использования



Приложение В

Диаграмма классов анализа



Приложение Г

Диаграмма проектных классов

cd ProjectClasses

