



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВУ ПРОЕКТУ

НА ТЕМУ:

**Информационная экспертная система по подбору
диеты**

Студент ИУ5-24М
(Группа)

(Подпись, дата)

Н.В. Журавлев
(И.О.Фамилия)

Руководитель

(Подпись, дата)

М.В. Виноградова
(И.О.Фамилия)

Консультант

(Подпись, дата)

(И.О.Фамилия)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ5
(Индекс)
В.И. Терехов
(И.О.Фамилия)
« ____ » _____ 20 24 г.

З А Д А Н И Е
на выполнение курсового проекта

по теме Технологии разработки программного обеспечения

Студент группы ИУ5-24М

Журавлев Николай Вадимович
(Фамилия, имя, отчество)

Тема курсового проекта Информационная экспертная система по подбору диеты

Направленность КП (учебная, исследовательская, практическая, производственная, др.)

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения проекта: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 15 нед.

Задание Выполнить разработку СОИУ в соответствии с описанием ее функциональности (определяется вариантом) на основе моделей унифицированного процесса (RUP). Написать программу, реализующую фрагмент СОИУ, и реализовать в ней паттерны бизнес-логики – transaction script, работы с БД – table data gateway и gof – итератор

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « 09 » февраля 20 24 г.

Руководитель курсового проекта

М.В. Виноградова
(Подпись, дата) (И.О.Фамилия)

Студент

Н.В. Журавлев
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Постановка задачи.....	3
Глава 1. Этап анализа и планирования требований.....	4
1.1. Перечень функциональных и нефункциональных требований.....	4
1.2 Модель предметной области.....	5
1.3 Выявленные актеры.....	5
1.4 Выявленные прецеденты, их приоритеты и описание.....	5
1.5 Диаграмма основных прецедентов.....	6
1.6 Перечень критических рисков и рекомендации по управлению.....	7
1.7 Перечень экранных форм и их сложность.....	7
1.8 Экспертные оценки скорости разработки и масштабных факторов; затраты, длительность и стоимость разработки.....	9
Глава 2. Этап проектирования.....	15
2.1 Уточненная диаграмма прецедентов.....	16
2.2 Описания прецедентов.....	16
2.3 Прототип пользовательского интерфейса.....	18
2.4 Диаграммы классов анализа: граничных, управляющих, сущностей.....	20
2.5 Диаграммы взаимодействия для основных прецедентов.....	21
2.6 Пакеты анализа и сервисные пакеты в форме обобщенной диаграммы классов.....	27
2.7 Распределение классов проектирования по подсистемам.....	27
2.8 Трассировка пакетов в подсистемы, классов анализа в классы проектирования.....	28
2.9 Диаграмма уровней подсистем.....	30
2.10 Диаграмма размещения подсистем.....	30
2.11 Трассировка подсистем в компоненты, классов проектирования в исходные файлы.....	31
2.12 Зависимость компонентов от исходных файлов.....	32
2.13 Перечень и состав итераций следующего этапа с указанием прецедентов, коопераций, пакетов, подсистем и компонентов для разработки.....	33
Глава 3. Этап построения.....	35
3.1 Экранные формы работающей программы.....	35
3.2 Исходный код программы, реализующей архитектурно-значимые прецеденты и паттерны.....	37
3.3 Полная диаграмма классов реализации.....	47
3.4 Диаграммы последовательностей для иллюстрации работы паттернов.....	48
3.5 Перечень и последовательность проведения тестов.....	54
Глава 4. Этап внедрения.....	59
4.1 Перечень программ и рекомендации по установке.....	59
4.2 Перечень документации для пользователей и заказчиков.....	60
4.3 Рекомендации по внедрению.....	60
Заключение.....	61

Постановка задачи

Выполнить разработку СОИУ в соответствии с описанием ее функциональности (определяется вариантом) на основе моделей унифицированного процесса (RUP). Написать программу, реализующую фрагмент СОИУ, и реализовать в ней паттерны бизнес-логики – transaction script, работы с БД - table data gateway и gof –итератор.

Глава 1. Этап анализа и планирования требований

Этап анализа и планирования требований - это важный этап в разработке программного обеспечения. Он включает в себя несколько ключевых шагов:

- Определить область применения СОИУ (предназначение, границы, интерфейсы с внешней средой, критерии сдачи-приемки);
- Определить прецеденты, критические для системы (основные функции и главные решения);
- Определить основные элементы архитектуры (для выполнения основного сценария);
- Определить и оценить самые опасные риски (угрожающие успеху разработки), предложить способы управления ими;
- Оценить затраты, длительность и стоимость разработки по модели COSOMO-2 этапа композиции приложения.

1.1. Перечень функциональных и нефункциональных требований

Спецификация основных проектных требований, ключевых характеристик и главных ограничений (перечень функциональных и нефункциональных требований).

В рамках ПО «Информационная экспертная система по подбору диеты» должны быть реализованы следующие функциональные требования:

- В ПО должна быть возможность регистрации пользователя;
- В ПО должна быть возможность создать новые диеты;
- В ПО должна быть возможность у пользователя указать его рост, вес, возраст;
- В ПО должна быть возможность подобрать диету пользователя в зависимости от его роста, веса, возраста;

В рамках ПО «Информационная экспертная система по подбору диеты» должны быть реализованы следующие нефункциональные требования:

- Дизайн ПО должен быть лаконичен и понятен любому пользователю;

- ПО должно иметь возможность запускаться на ОС Windows;
- Время подбора диеты должно не превышать 5 секунд;

1.2 Модель предметной области

Модель предметной области представлена на рис. 1.

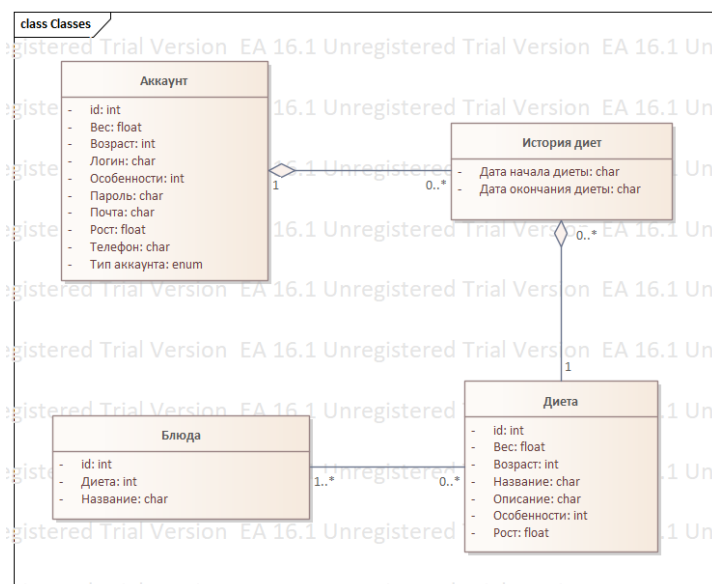


Рисунок 1 – Диаграмма классов предметной области

1.3 Выявленные актеры

Всего можно выявить 2 актёра:

- 1) Пользователь
 - a. Выбирает диету
 - b. Может авторизоваться
 - c. Заполняет данные о себе
- 2) Диетолог
 - a. Добавляет диету
 - b. Добавляет блюда

1.4 Выявленные прецеденты, их приоритеты и описание

Авторизация (высокий приоритет) – авторизация пользователя в системе.

Подбор диеты (высокий приоритет) – подбор диеты для пользователя.

Создание диеты (высокий приоритет) – создание диеты диетологом.

Регистрация (средний приоритет) – регистрация диетолога и пользователя в системе.

Заполнение данных о себе (средний приоритет) – пользователь заполняет данные о себе, которые нужны для подбора диеты.

Добавление блюда (средний приоритет) – создание блюда в диете диетологом.

Получение истории диет (низкий приоритет) – получение истории диет, которые когда-либо были у пользователя.

Восстановление пароля (низкий приоритет) – восстановление пароля у любого пользователя.

1.5 Диаграмма основных прецедентов

Диаграмма основных прецедентов представлена на рис. 2.

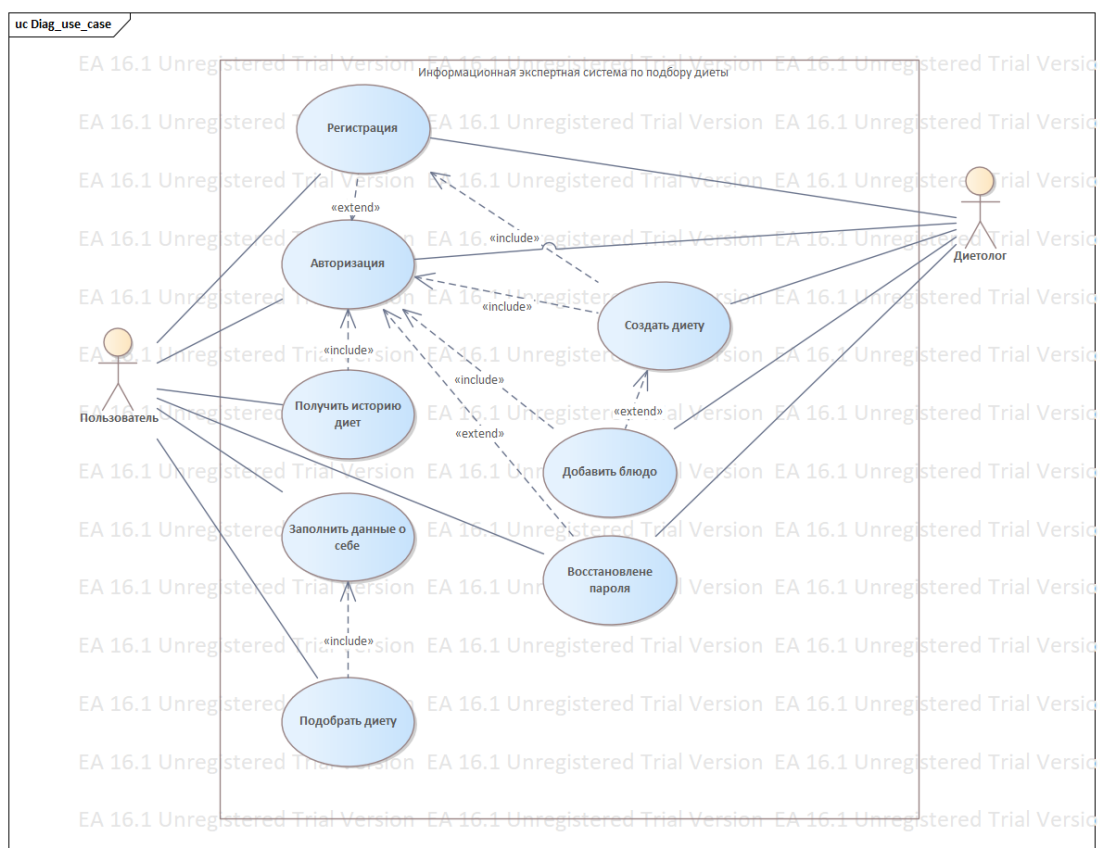


Рисунок 2 – Диаграмма основных прецедентов

1.6 Перечень критических рисков и рекомендации по управлению

Таблица 1 – Критические риски и рекомендации по их устранению

Риски	План управления
Критическая программная ошибка	Иметь людей, которые могут в кратчайшие сроки устранить ошибку
СУБД теряет данные	Делать бэкапы для возможности восстановления в определённый промежуток
Разработка неверного алгоритма подбора диеты	Уточнять алгоритм у специалистов
Отставание по срокам	Привлечение дополнительной рабочей силы
Потеря финансирования	Поиск новых инвесторов
Дефицит процессорной памяти	Добавление аппаратных мощностей
Разработка неверного пользовательского интерфейса	Провести GUI тестирование и исправить выявленные проблемы

1.7 Перечень экранных форм и их сложность

Были выявлены следующие экранные формы:

1) Форма авторизации

а. Функция авторизации

1. Поле логин
2. Поле пароля
3. Кнопка авторизация

б. Функция восстановления пароля

1. Поле логин
2. Кнопка восстановления пароля

2) Форма подборы диеты

а. Функция подборы диеты

1. Кнопка подобрать диету
2. Поле результата

3) Форма получения истории диет

- а. Функция получения истории диет
 - 1. Кнопка получить историю диет
 - 2. Поле результата
- 4) Форма заполнения данных о себе
 - а. Функция заполнения данных о себе
 - 1. Поле пароль
 - 2. Поле возраст
 - 3. Поле вес
 - 4. Поле рост
 - 5. Поле особенности
 - 6. Кнопка добавить
- 5) Форма добавления блюда
 - а. Функция добавление блюда
 - 1. Поле название
 - 2. Поле К какой диете принадлежит
 - 3. Кнопка добавить
- 6) Форма добавления диеты
 - а. Функция добавления диеты
 - 1. Поле названия
 - 2. Поле возраст
 - 3. Поле вес
 - 4. Поле рост
 - 5. Поле особенности
 - 6. Поле описание

7. Кнопка добавить

1.8 Экспертные оценки скорости разработки и масштабных факторов; затраты, длительность и стоимость разработки

В проекте “Информационная экспертная система по подбору диеты” можно выделить следующие функции:

- 1) Подбор диеты;
- 2) База данных диет и способы взаимодействия с ней;
- 3) База данных пользователей и способы взаимодействие с ней.

Обозначим их P_i , где i – номер функции.

В качестве базиса используется таблица аналогов из книги Орлова С.А.

Результат представлен в таб.2.

Таблица 2. Расчёт затрат, стоимости и LOC по аналогам

Проект	Затраты, чел.-мес.	Стоимость, тыс. руб	KLOC, тыс. LOC	LOC
P_1	0,25	5,357	0,3	300
P_2	0,25	3,571	0,2	200
P_3	0,25	3,571	0,2	200

Функция P_1 соответствует аналогу “aaa01”. P_2 - “bbb02”. P_3 -“ccc03”.

Расчет средней производительности и стоимости по аналогам, оценка затрат и стоимости

Стоимость по аналогам рассчитана в предыдущем пункте в таб.2.

Для расчёта средней производительности посчитаем $ПРОИЗВ_i$ – производительность i -ой функции:

$$ПРОИЗВ_1 = \frac{KLOC_1}{ЗАТРАТЫ_1} = 1.2$$

$$\text{ПРОИЗВ}_2 = \frac{KLOC_2}{\text{ЗАТРАТЫ}_2} = 0.8$$

$$\text{ПРОИЗВ}_3 = \frac{KLOC_3}{\text{ЗАТРАТЫ}_3} = 0.8$$

Средняя произвольность рассчитаем по следующей формуле:

$$\text{ПРОИЗВ}_{\text{ср}} = \frac{\text{ПРОИЗВ}_1 + \text{ПРОИЗВ}_2 + \text{ПРОИЗВ}_3}{3} = 0,933333333$$

Так же необходимо посчитать среднюю удельную стоимость, для этого посчитаем ПРОИЗВ_i – производительность i-ой функции:

$$\text{УД_СТОИМОСТЬ}_1 = \frac{\text{СТОИМОСТЬ}_1}{KLOC_1} = 17,85666667$$

$$\text{УД_СТОИМОСТЬ}_2 = \frac{\text{СТОИМОСТЬ}_2}{KLOC_2} = 17,855$$

$$\text{УД_СТОИМОСТЬ}_3 = \frac{\text{СТОИМОСТЬ}_3}{KLOC_3} = 17,855$$

Средняя удельная стоимость рассчитаем по следующей формуле:

$$\text{УД_СТОИМОСТЬ}_{\text{ср}} = \frac{\text{УД_СТОИМОСТЬ}_1 + \text{УД_СТОИМОСТЬ}_2 + \text{УД_СТОИМОСТЬ}_3}{3} = 17,8(5)$$

Для каждой функции рассчитаем вычисляем LOC-оценки по формуле:

$$LOC_{\text{ож}i} = (LOC_{\text{лучш}i} + LOC_{\text{худш}i} + 4 * LOC_{\text{вероятн}i}) / 6$$

Получившийся результат:

$$LOC_{\text{ож}1} = \frac{(8000 + 14000 + 4 * 10000)}{6} = 308, (3)$$

$$LOC_{\text{ож}2} = \frac{(15000 + 23000 + 4 * 20000)}{6} = 208, (3)$$

$$LOC_{\text{ож}3} = \frac{(25000 + 29000 + 4 * 27000)}{6} = 200$$

Оценка затрат считается следующим образом:

$$\text{ЗАТРАТЫ} = \left(\sum LOC_{\text{ож}i} \right) / \text{ПРОИЗВ}_{\text{ср}} = 767,8571429$$

Оценка стоимости считается следующим образом:

$$\text{СТОИМОСТЬ} = \left(\sum \text{LOC}_{\text{ож}i} \right) * \text{УД_СТОИМОСТЬ}_{\text{ср}} = 12796,48148$$

Расчет затрат

Для расчёта затрат используется формула:

$$\text{ЗАТРАТЫ} = A * \text{РАЗМЕР}^B * M_e + \text{ЗАТРАТЫ}_{\text{auto}}$$

где:

- Масштабный коэффициент $A = 2,5$;
- Показатель B отражает нелинейную зависимость затрат от размера проекта (размер системы РАЗМЕР выражается в тысячах LOC);
- Множитель поправки M_e зависит от 7 формирователей затрат, характеризующих продукт, процесс и персонал;
- Слагаемое $\text{ЗАТРАТЫ}_{\text{auto}}$ отражает затраты на автоматически генерируемый программный код.

Значение показателя степени B изменяется в диапазоне 1,01... 1,26, зависит от пяти масштабных факторов W_i и вычисляется по формуле

$$B = 1,01 + 0,01 \sum W_i$$

Таблица 3. Масштабные факторы

Проект	PREC	FLEX	RESL	TEAM	PMAT
П ₁	4	4	4	0	3
П ₂	1	4	1	1	3
П ₃	3	4	1	0	3

На основе оценки для каждого формирователя по таблице Бозма определяется множитель затрат EM_i .

Перемножение всех множителей затрат формирует множитель поправки:

$$M_e = \prod_{i=1}^7 EM_i$$

Таблица Бозма, по которой определяются множители затрат:

Таблица 4. Таблица Бозма

	0	1	2	3	4	5	6
PERS	1,33	1,22	1,11	1	0,89	0,78	0,67
RCPX	0,67	0,78	0,89	1	1,11	1,22	1,33
RUSE	0,67	0,78	0,89	1	1,11	1,22	1,33
PDIF	0,67	0,78	0,89	1	1,11	1,22	1,33
PREX	1,33	1,22	1,11	1	0,89	0,78	0,67
FCIL	1,33	1,22	1,11	1	0,89	0,78	0,67
SCED	1,33	1,22	1,11	1	1	1	1

Для каждого формирователя затрат определим оценку и занесём множители в таблицу:

Таблица 5. Множители затрат

Проект	PERS	RCPX	RUSE	PDIF	PREX	FCIL	SCED
П ₁	0,78	1	1,33	0,78	1,11	0,78	1
П ₂	0,78	1	1,33	0,78	1	0,78	1
П ₃	0,78	1	1,33	0,78	1	0,78	1

Слагаемое $ЗАТРАТЫ_{auto} = 0$, так как автогенерируемый код отсутствует.

По итогу вычисления затрат получается:

Таблица 6. Подсчёт затрат

Проект	A	РАЗМЕР	B	M_e	$ЗАТРАТЫ_{auto}$	ЗАТРАТЫ
П ₁	2,5	10	1,16	0,7005811176	0	0,433370247

П ₂	2,5	20	1,11	0,63115416	0	0,264374007
П ₃	2,5	27	1,12	0,63115416	0	0,260153128

Расчет длительности и стоимости разработки

Описание длительности вычисляется следующим образом:

$$(TDEV) = [3,0 * ЗАТРАТЫ^{(0,33+0,2(B-1,01))}] * SCEDPercentage/100 \text{ [мес]},$$

Где:

- Значение показателя степени В изменяется в диапазоне 1,01... 1,26, зависит от масштабных факторов W_i и вычисляется по формуле выше
- SCEDPercentage - процент увеличения (уменьшения) номинального графика.

Так как нужно определить номинальный график, то SCEDPercentage = 100.

После расчёта длительности получаются следующие результаты:

Таблица 6. Расчёт длительности

Проект	(TDEV)
П ₁	2,220191971
П ₂	1,883205952
П ₃	1,867591468

Стоимости проекта рассчитывается по формуле:

$$\text{СТОИМОСТЬ} = \text{ЗАТРАТЫ} \times \text{РАБ_КОЭФ},$$

где РАБ_КОЭФ = 1

После расчёта стоимости получаем следующие результаты:

Таблица 7. Расчёт стоимости

Проект	Стоимость
Π_1	0,433370247
Π_2	0,264374007
Π_3	0,260153128

Глава 2. Этап проектирования

Этап проектирования включает в себя несколько ключевых этапов, которые играют важную роль в разработке различных продуктов и систем. Эти этапы включают в себя:

1. Составить модель требований (выделить актеров и прецеденты; создать прототип пользовательского интерфейса; детализировать и структурировать прецеденты).
2. Составить модель анализа (анализ архитектуры - выделить пакеты анализа и сервисные пакеты, определить классы сущностей и общие специальные требования; анализ прецедентов — определить классы анализа и их взаимодействие; анализ классов — определить ответственности, атрибуты и связи классов; анализ пакетов – определить состав и зависимости пакетов).
3. Составить модель проектирования (определить узлы и сетевые конфигурации, подсистемы и интерфейсы между ними, архитектурно-значимые и активные классы, обобщенные механизмы проектирования).
4. Выбрать и обосновать структурные шаблоны проектирования (MVC, РСМЕФ и т. д.)
5. Реализовать базовый уровень архитектуры на основе модели проектирования с применением паттернов базы данных и бизнес-логики.
6. Отслеживать риски, устранить наиболее серьезные.
7. Составить план итераций следующего этапа.

2.1 Уточненная диаграмма прецедентов

Уточненная диаграмма прецедентов представлена на рис. 3.

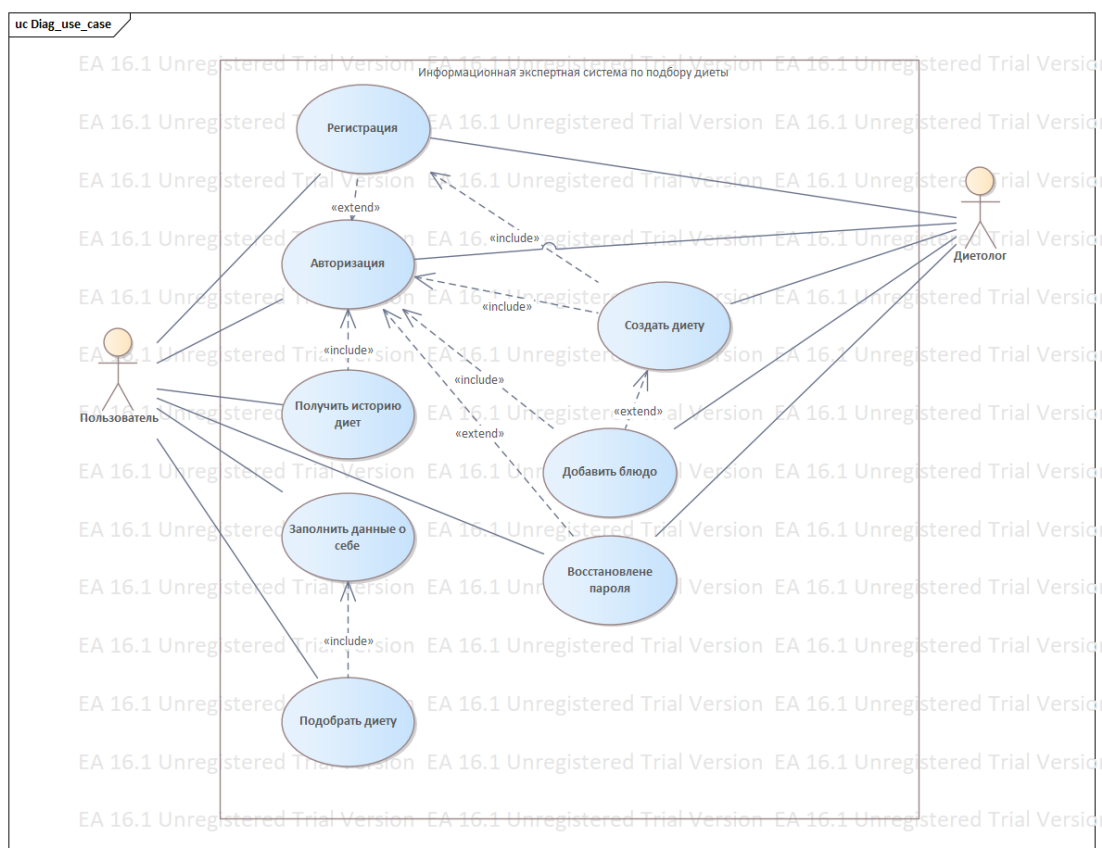


Рисунок 3 – Диаграмма основных прецедентов

2.2 Описания прецедентов

Таблица 7 – Описание прецедентов

Прецедент: «Аторизация»
Краткое описание: Авторизация человека в системе
Главные актёры: Пользователь, Диетолог
Второстепенные актёры: Нет
Предусловие: 1. Пользователь должен загрузить систему
Основной поток: 1. Прецедент начинается после захода пользователя на сайт 2. Проверяется тип запроса, если тип “POST”, то получения данных из заполненной формы (иначе A1). 3. Происходит подключение к базе данных 4. Получение из базы данных информации при поиске в БД используется логин 5. Выполняется проверка пароля на равенство введённому (если не находит, то A2) 6. В сессию сохраняется логин пользователя 7. Происходит переадресация на основную страницу
Постусловие: Пользователь попадает на основную страницу системы

Продолжение таблицы 7

<p>Альтернативные потоки:</p> <p>Альтернативный поток А1:</p> <p>2. Проверяется тип запроса, если тип “GET”, то выдаётся форма для авторизации (иначе основной поток)</p> <p>Альтернативный поток А2:</p> <p>5. Выполняется проверка пароля на равенство введённому (если равны, то основной поток)</p> <p>6. Выдаётся форма авторизации с сообщением о неверном логике или пароле</p>
Прецедент: «Создание диеты»
Краткое описание: Диетолог создаёт диету, которые могут быть предоставлены пользователю
Главные актёры: Диетолог
Второстепенные актёры: Нет
<p>Предусловие:</p> <p>1. Пользователь должен загрузить систему</p> <p>2. Пользователь должен быть авторизован</p>
<p>Основной поток:</p> <p>1. Прецедент начинается после нажатия кнопки добавление диеты</p> <p>2. Проверяется тип запроса, если тип “POST”, то получения данных из заполненной формы (иначе А1).</p> <p>3. Происходит подключение к базе данных</p> <p>3. В базу данных в таблицу диет добавляется диета с полученными из формы данными</p> <p>4. Происходит переадресация на основную страницу</p>
Постусловие: сообщение об успешном создании
<p>Альтернативные потоки:</p> <p>2. Проверяется тип запроса, если тип “GET”, то выдаётся форма для заполнения данных о добавляемых диетах (иначе основной поток)</p>
Прецедент: «Подбор диеты»
Краткое описание: подбор диеты для пользователя
Главные актёры: Пользователь
Второстепенные актёры: Нет
<p>Предусловие:</p> <p>1. Пользователь должен загрузить систему</p> <p>2. Пользователь должен быть авторизован</p> <p>3. Пользователь должен заполнить информацию о себе</p>
<p>Основной поток:</p> <p>1. Прецедент начинается после нажатия кнопки подобрать диету</p> <p>2. Выполняется подключение к БД</p> <p>3. Из сессии получается логин пользователя</p> <p>4. По полученному логину получается основная информация об аккаунте.</p> <p>5. Система в соответствии с заполненной информации пользователя о себе выбирает из БД информацию о диете с условием, что особенности совпадают, а остальные данные имеют разницей в 5 значений</p> <p>6. Если такие диеты есть, то система сохраняет в БД диету в историю для конкретного пользователя (иначе А1)</p> <p>7. Для полученной диеты из БД выбираются все блюда, которые к ней относятся</p> <p>8. Пользователю выдаётся форма с найденной диетой и блюдами</p>

Продолжение таблицы 7

Постусловие: сообщение с информации о диете
Альтернативные потоки: Альтернативный поток А1: 6. Если таких диеты нет, то пользователю выдаётся форма с сообщением о не нахождении диеты (иначе основной поток)

2.3 Прототип пользовательского интерфейса

Интерфейсы функции о себе, авторизации и добавления диеты представлены на рис.4, рис.5, рис.6, соответственно.

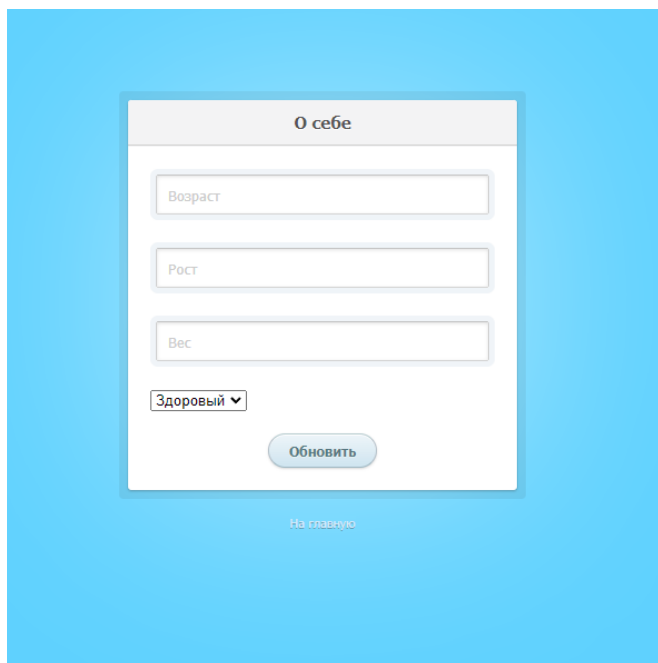


Рисунок 4 – Интерфейс страницы для заполнения информации о себе

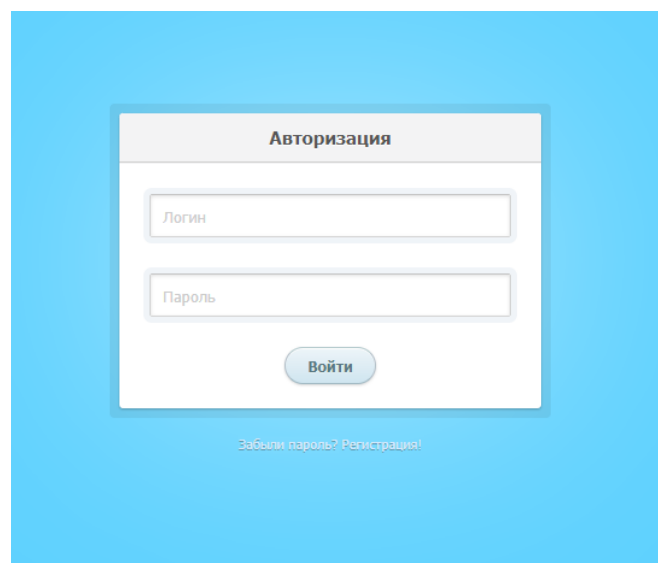


Рисунок 5 – Интерфейс страницы для авторизации

Добавление диеты

Название

Описание

Возраст

Рост

Вес

Здоровый ▼

Добавить

[На главную](#)

Рисунок 6 – Интерфейс страницы для добавления диеты

2.4 Диаграммы классов анализа: граничных, управляющих, сущностей

Диаграмма классов сущностей представлена на рис. 7.

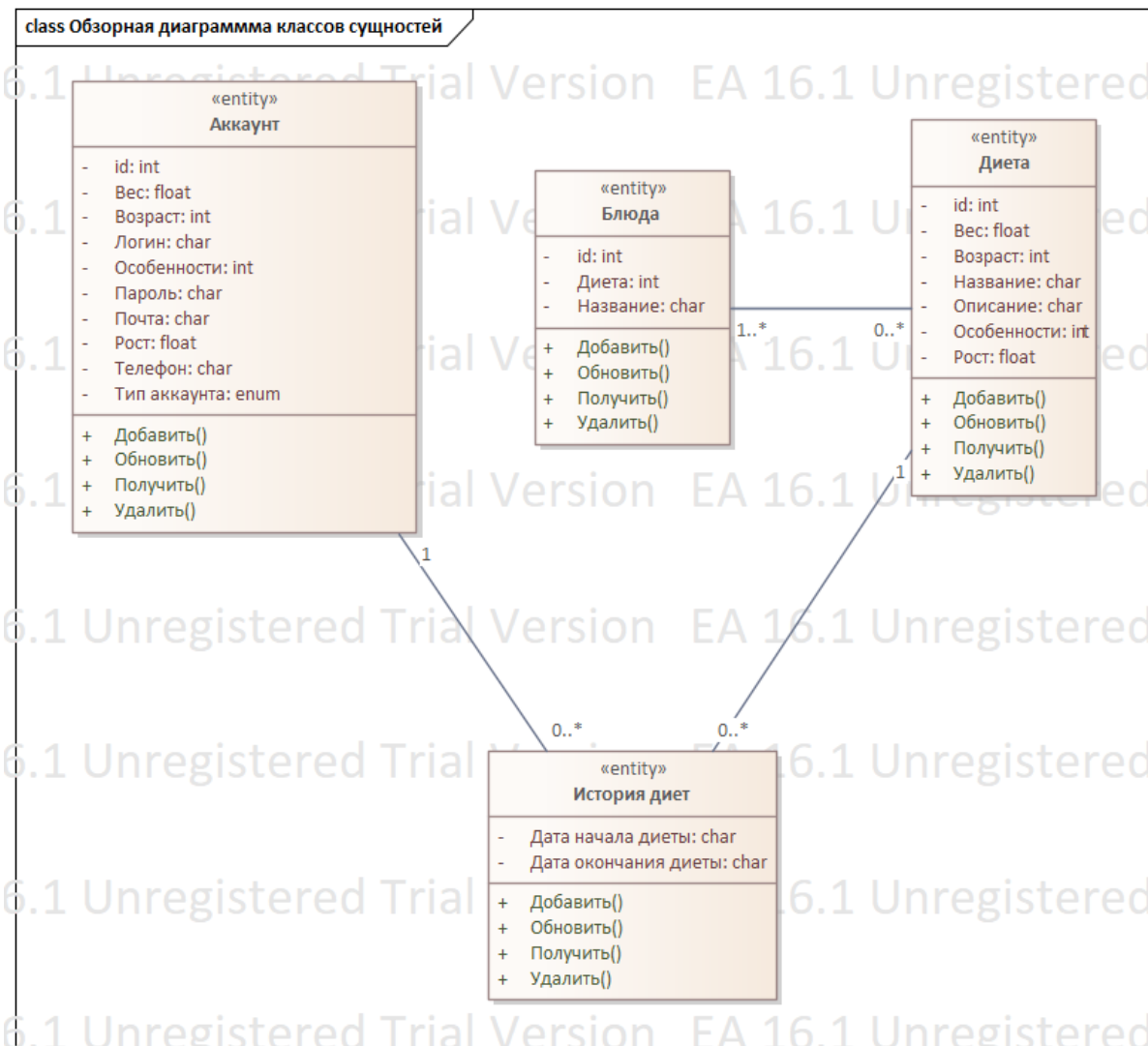


Рисунок 7 – Диаграмма классов сущностей

Диаграмма управляющих классов представлена на рис. 8.

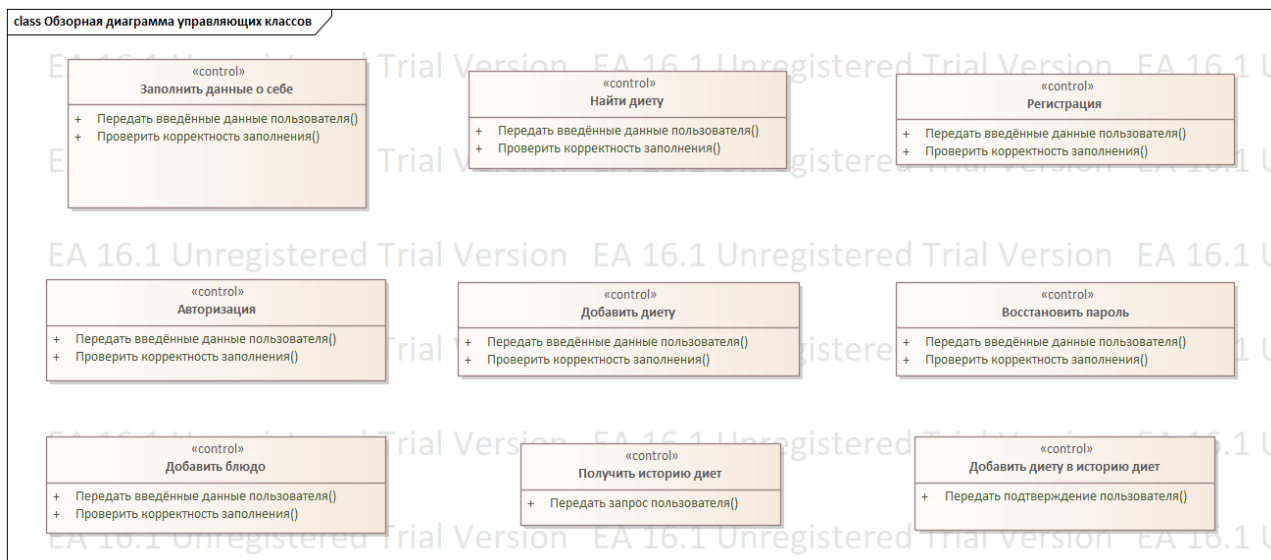


Рисунок 8 – Диаграмма управляющих классов

Диаграмма граничных классов представлена на рис. 9.



Рисунок 9 – Диаграмма граничных классов

2.5 Диаграммы взаимодействия для основных прецедентов

Диаграмма классов для прецедента авторизация представлена на рис. 10.

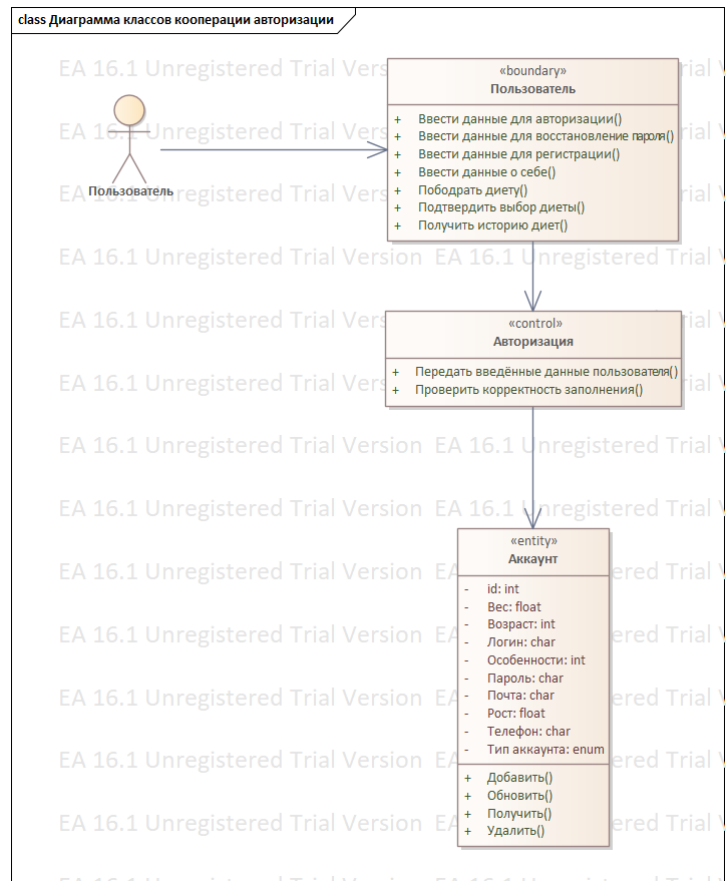


Рисунок 10 – Диаграмма классов для авторизации

Диаграмма последовательностей для прецедента авторизация представлена на рис. 11.

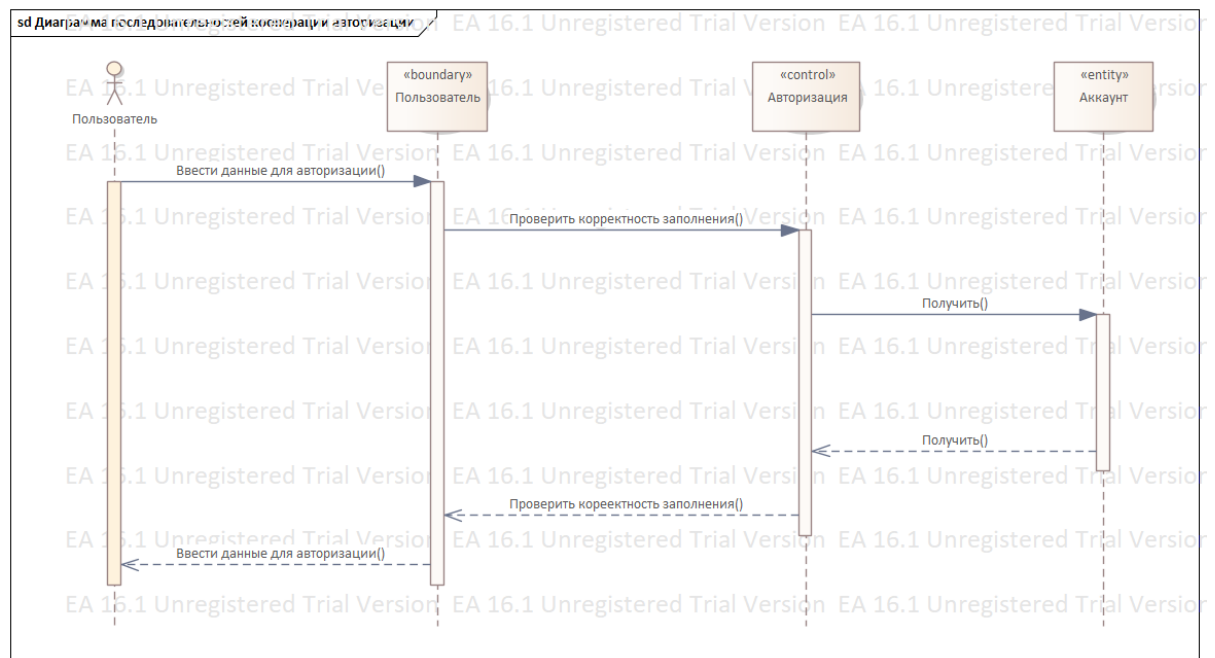


Рисунок 11 - Диаграмма последовательностей для прецедента авторизация

Диаграмма классов для прецедента заполнение характеристик человека представлена на рис. 12.

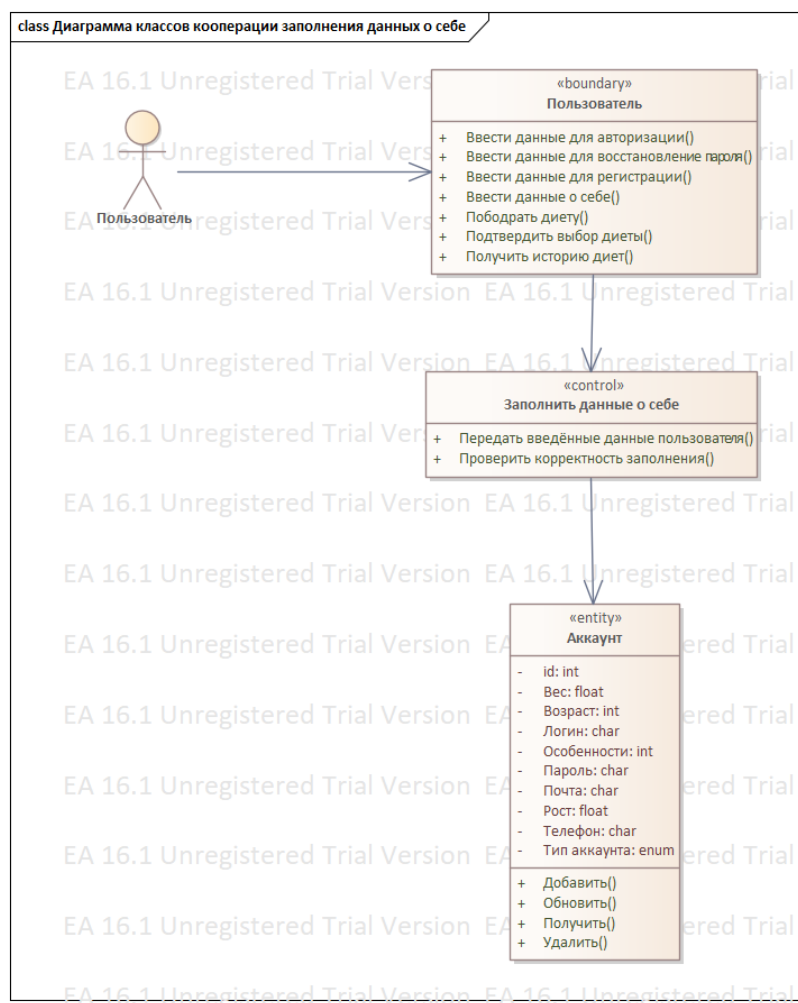


Рисунок 12 - Диаграмма классов для прецедента заполнение характеристик человека

Диаграмма последовательностей для прецедента заполнение характеристик человека представлена на рис. 13.

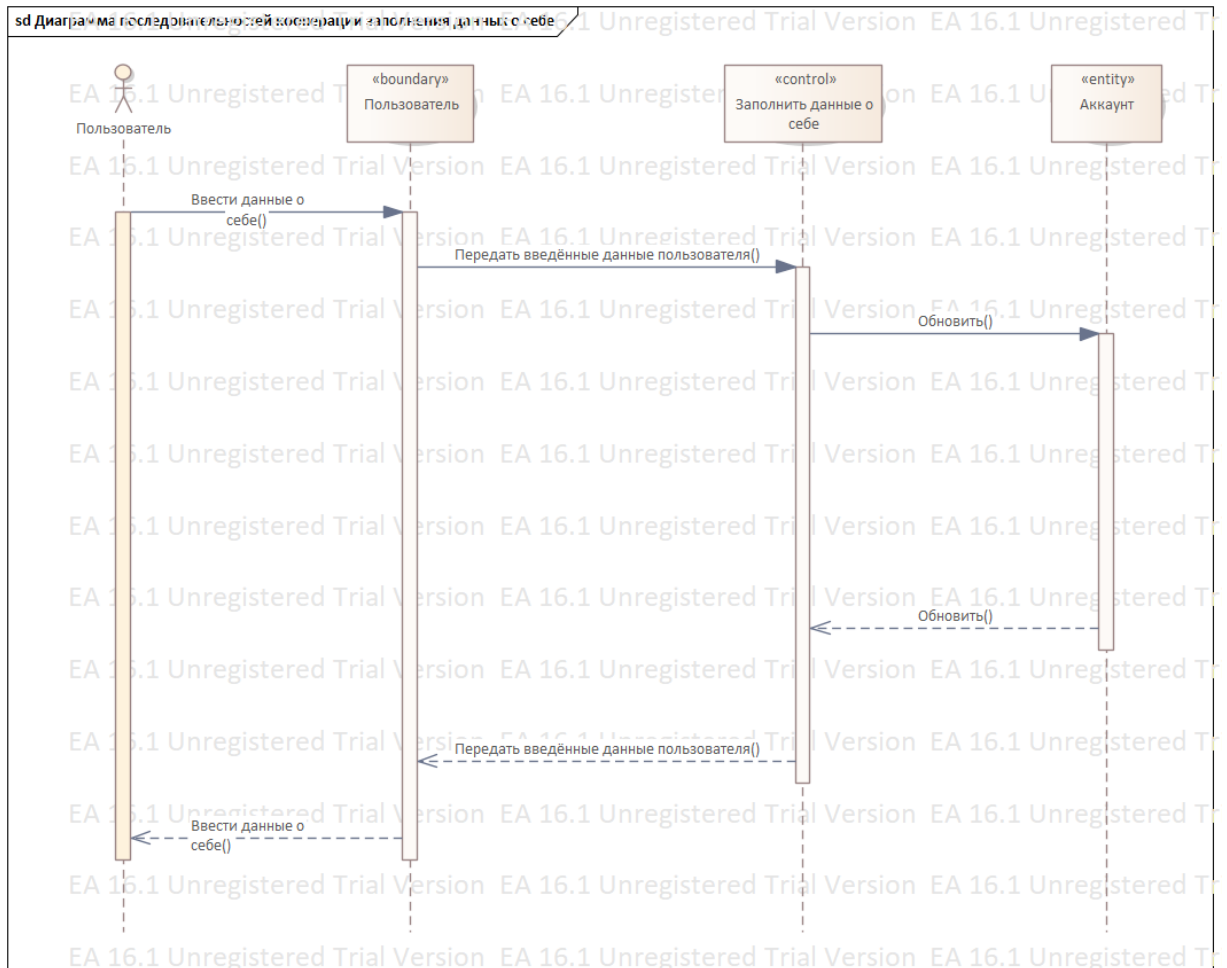


Рисунок 13 - Диаграмма последовательностей для прецедента заполнение характеристик человека

Диаграмма классов для прецедента подбор диеты представлен на рис. 14.

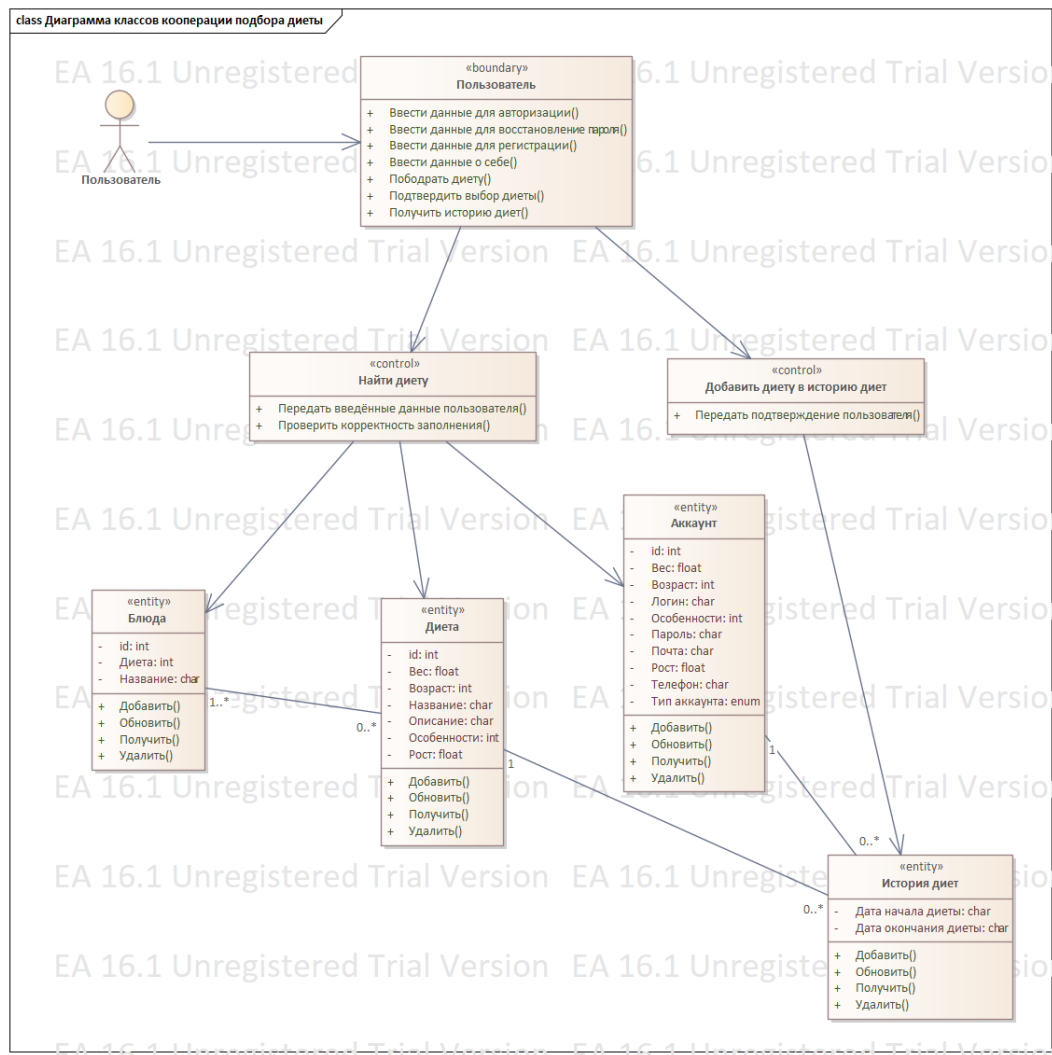


Рисунок 14 - Диаграмма классов для прецедента подбор диеты

Диаграмма последовательностей для прецедента подбор диеты
представлен на рис. 15.

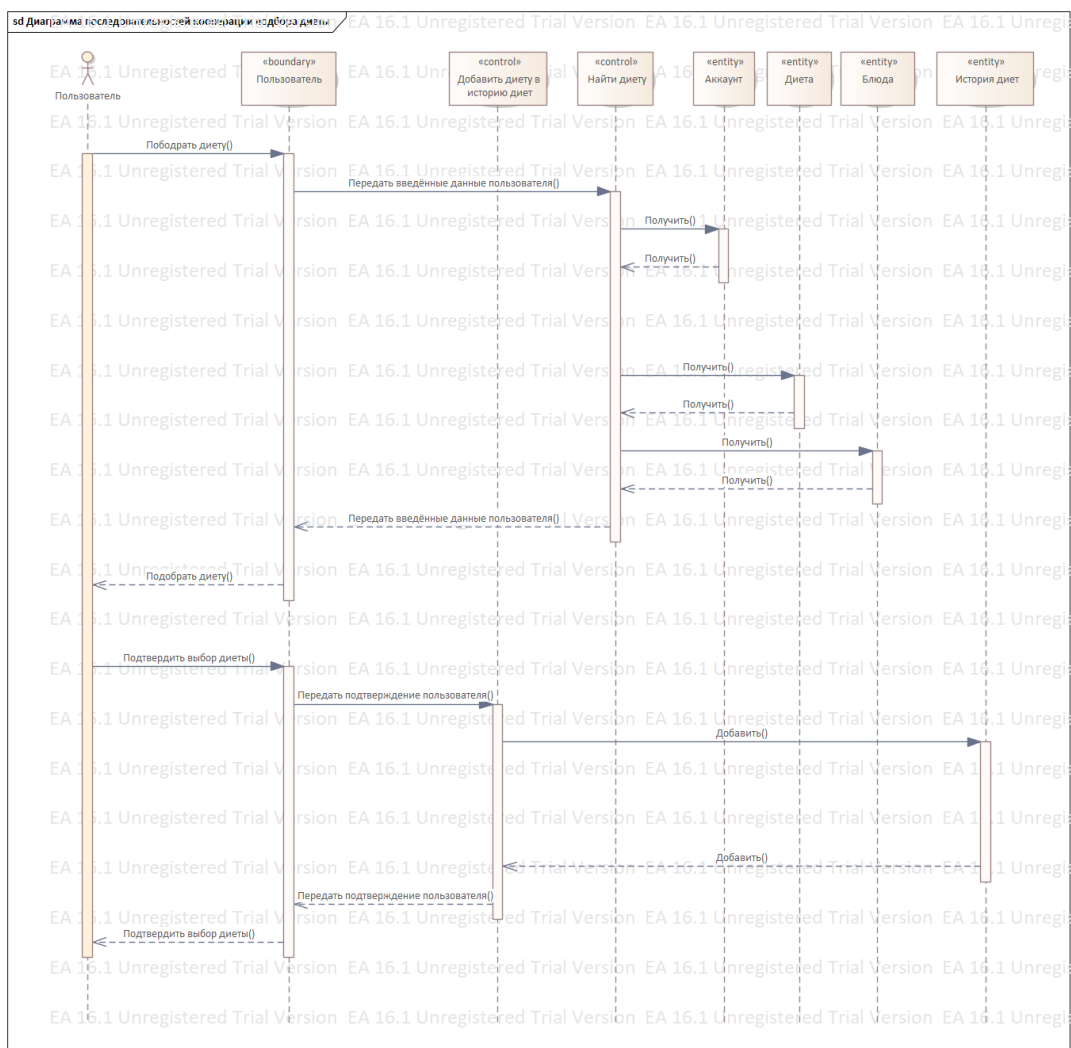


Рисунок 15 - Диаграмма последовательностей для прецедента подбор диеты

Пакеты анализа и сервисные пакеты в форме обобщенной диаграммы классов представлены на рис. 16.



Диаграмма распределения классов проектирования по подсистемам представлена на рис.17.

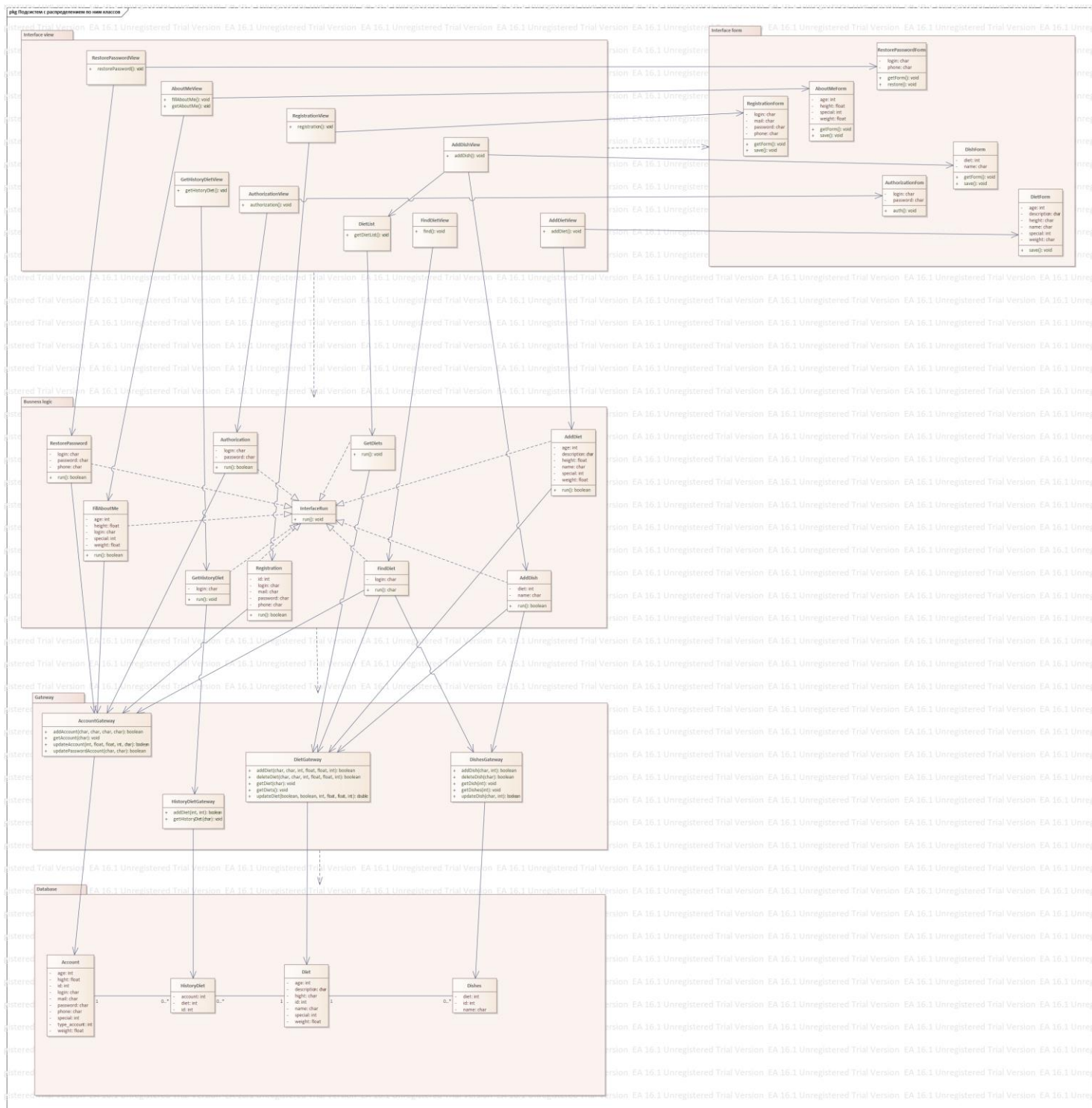


Рисунок 17 - Диаграмма распределения классов проектирования по подсистемам

2.8 Трассировка пакетов в подсистемы, классов анализа в классы проектирования

Трассировка классов анализа в подсистемы представлены на рис. 18.

2.9 Диаграмма уровней подсистем

Диаграмма уровней подсистем представлена на рис. 20.

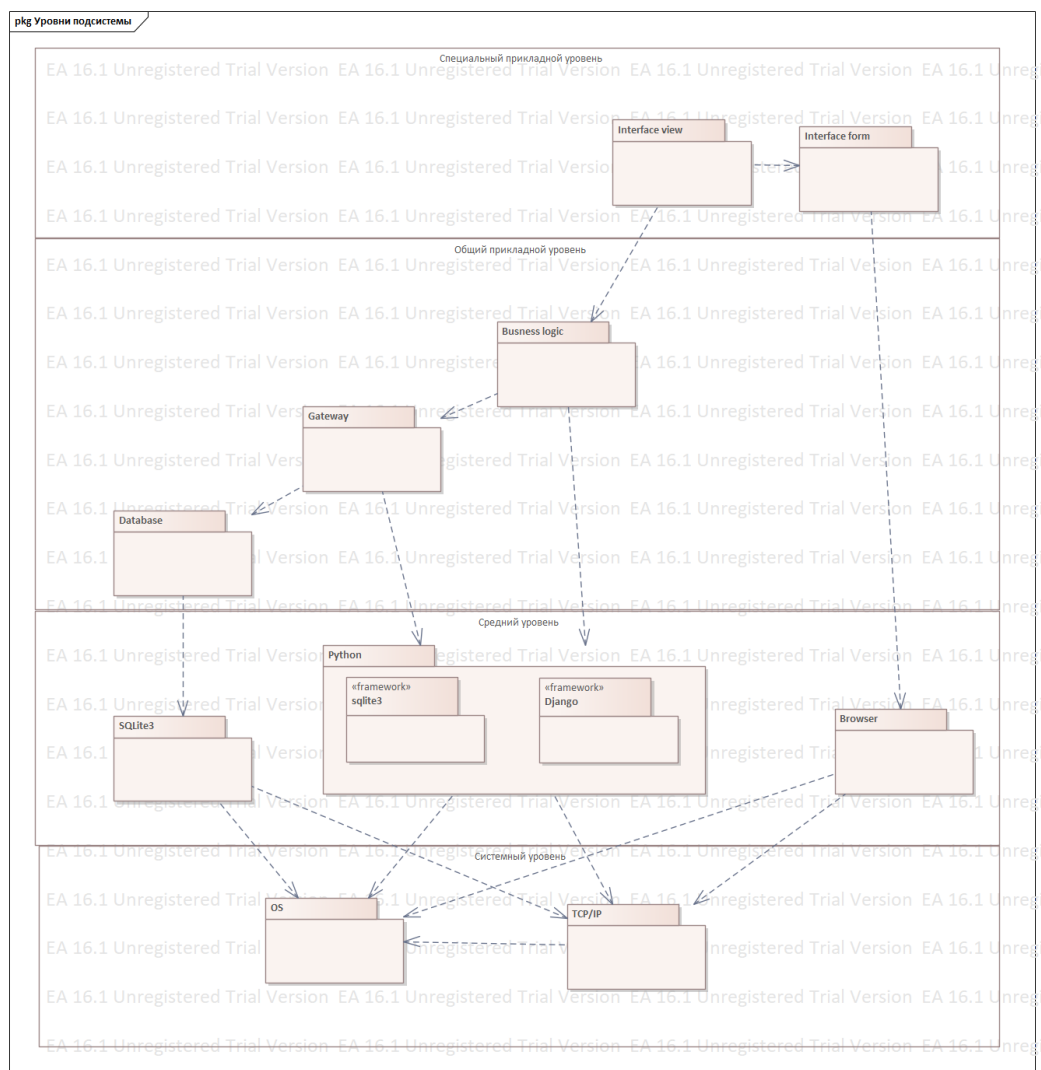


Рисунок 20 - Диаграмма уровней подсистем

2.10 Диаграмма размещения подсистем

Диаграмма размещения подсистем представлена на рис. 21.

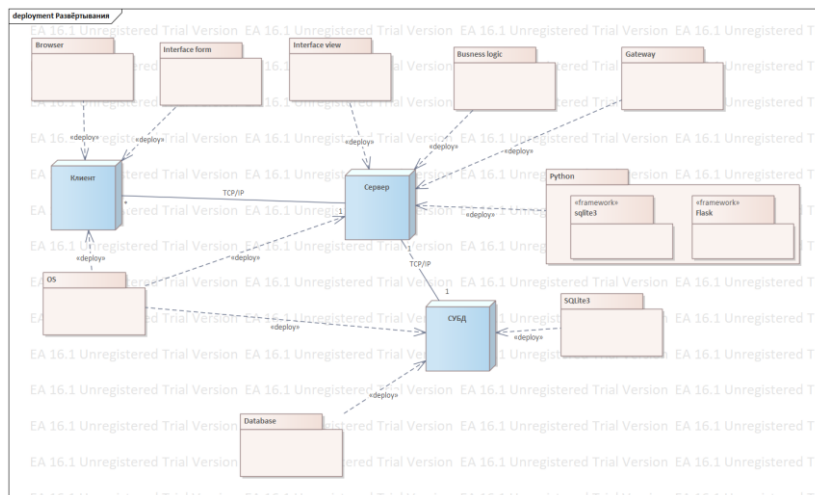


Рисунок 21 – Диаграмма размещения подсистем

2.11 Трассировка подсистем в компоненты, классов проектирования в исходные файлы

Трассировка подсистем в компоненты представлена на рис. 22.

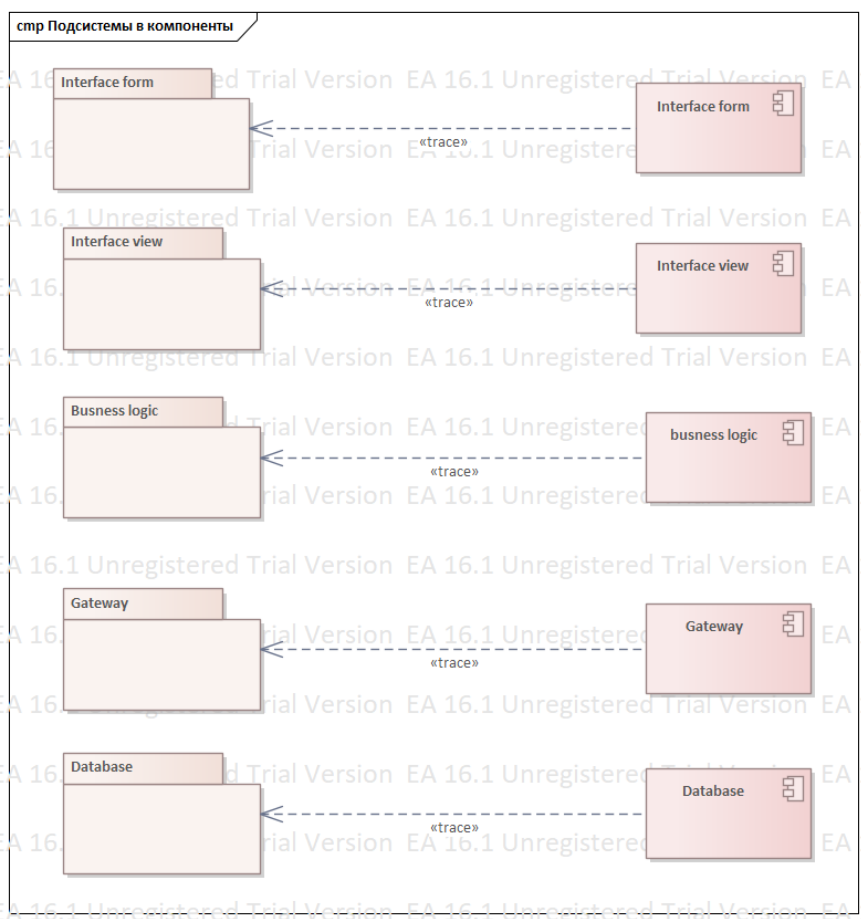


Рисунок 22 - Трассировка подсистем в компоненты

Трассировка классов проектирования в исходные файлы представлена на рис. 23.

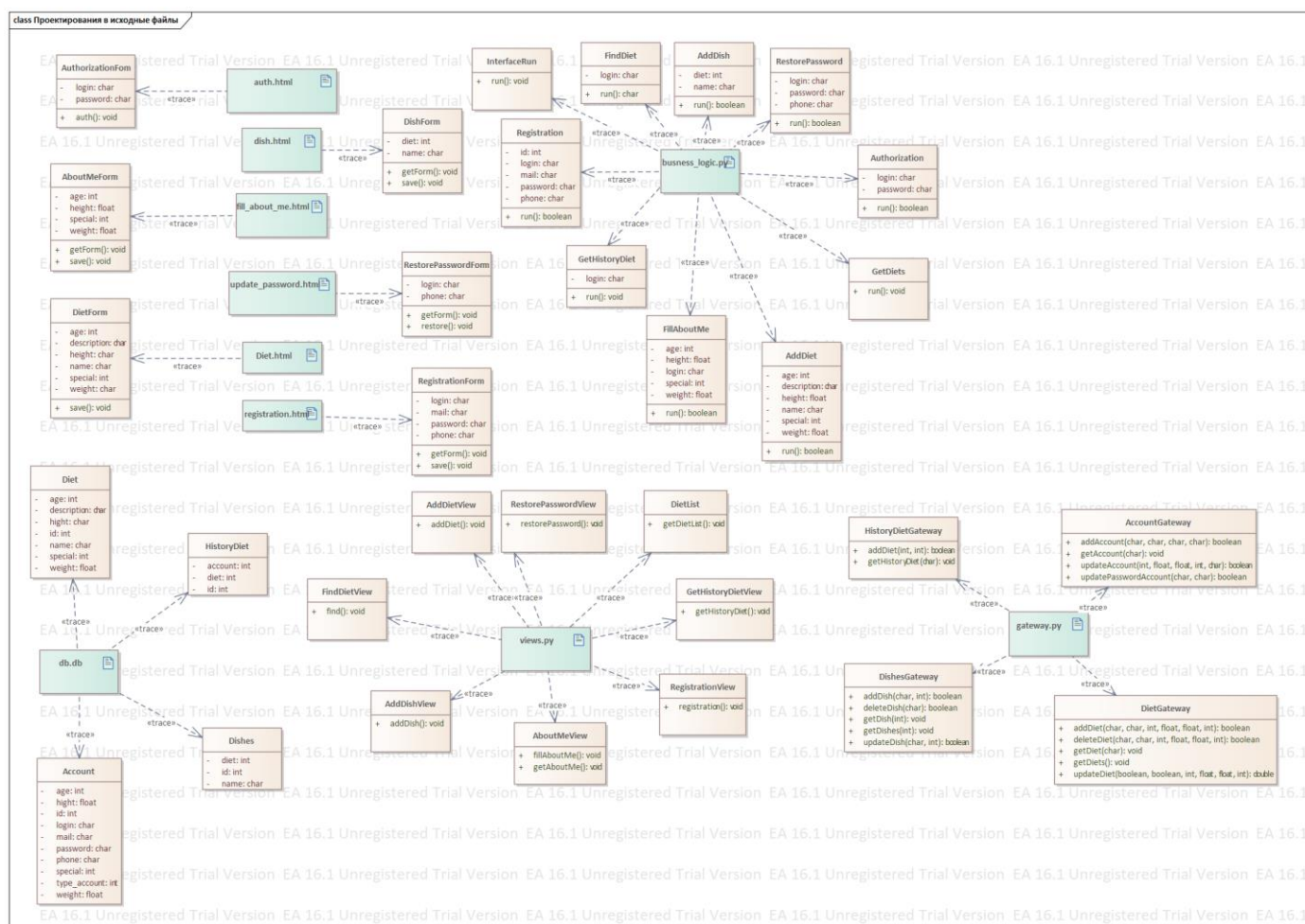


Рисунок 23 - Трассировка классов проектирования в исходные файлы

2.12 Зависимость компонентов от исходных файлов

Диаграмма зависимости компонентов от исходных файлов представлена на рис. 24.

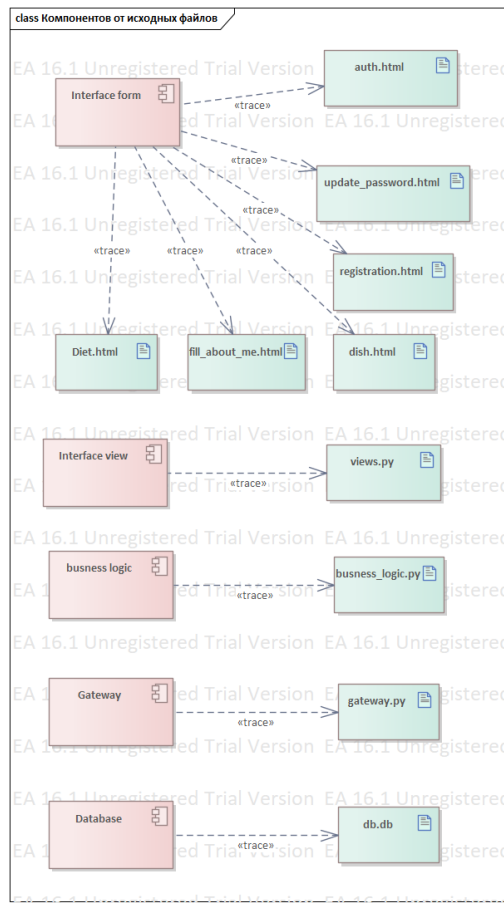


Рисунок 24 - Диаграмма зависимости компонентов от исходных файлов

2.13 Перечень и состав итераций следующего этапа с указанием прецедентов, коопераций, пакетов, подсистем и компонентов для разработки

Перечень и состав итераций указан в табл.8.

Таблица 8 – Перечень итераций

Итерации	Прецеденты	Пакеты	Подсистемы	Компоненты
Итерация 0	Получение диеты	Пакет пользователя	Interface	Interface
	Авторизация	Пакет авторизации	Business logic	Business logic
		Пакет данных	Gateway	Gateway
			Database	Database

Продолжение таблицы 8

Итерация 1	Заполнение данных о себе	Пакет пользователя (дополнен)	Interface (дополнен)	Interface (дополнен)
	Добавление диеты	Пакет диетолога	Business logic (дополнен)	Business logic (дополнен)
	Добавление блюда	Пакет данных (дополнен)	Gateway (дополнен)	Gateway (дополнен)
	Регистрация		Database (дополнен)	Database (дополнен)
Итерация 2	Восстановление пароля	Пакет пользователя (дополнен)	Interface (дополнен)	Interface (дополнен)
	Получение истории диет	Пакет данных (дополнен)	Business logic (дополнен)	Business logic (дополнен)
			Gateway (дополнен)	Gateway (дополнен)
			Database (дополнен)	Database (дополнен)

Глава 3. Этап построения

На этапе построения происходит реализация разработанной архитектуры и детального проектирования. Это включает в себя создание конечного продукта на основе предварительно определенных требований и разработанных планов. Важными аспектами этого этапа являются:

1. Полностью определить и реализовать прецеденты.
2. Завершить анализ, проектирование, реализацию и тестирование.
3. Применить паттерны бизнес-логики, работы с БД и Gof.
4. Провести тестирование и отладку

В данной курсовой работы используются следующие паттерны:

1. Transaction script для бизнес-логики
2. Table data gateway для работы с базой данных

3.1 Экранные формы работающей программы

Были выявлены следующие экранные формы:

1) Форма авторизации

а. Функция авторизации

1. Поле логин
2. Поле пароля
3. Кнопка авторизация

б. Функция восстановления пароля

1. Поле логин
2. Поле телефона
3. Поле нового пароля
4. Кнопка восстановления пароля

2) Форма подборки диеты

а. Функция подборки диеты

1. Кнопка подобрать диету
2. Поле результата

3) Форма заполнения данных о себе

а. Функция заполнения данных о себе

1. Поле пароль
2. Поле возраст
3. Поле вес
4. Поле рост
5. Поле особенности
6. Кнопка добавить

4) Форма добавления блюда

а. Функция добавление блюда

1. Поле название
2. Поле к какой диете принадлежит
3. Кнопка добавить

5) Форма добавления диеты

а. Функция добавления диеты

1. Поле названия
2. Поле возраст
3. Поле вес
4. Поле рост
5. Поле особенности
6. Поле описание
7. Кнопка добавить

3.2 Исходный код программы, реализующей архитектурно-значимые прецеденты и паттерны

Реализация классов Transaction script

Интерфейс с методом run, от которого наследуются все остальные классы сценариев транзакций:

```
class InterfaceRun:
    def run(self):
        pass
```

Класс Registration для выполнения сценария регистрации:

```
class Registration(InterfaceRun):
    def __init__(self, login, password, mail, phone, type_account, cursor):
        super().__init__()
        self.login = login
        self.password = password
        self.mail = mail
        self.phone = phone
        self.type_account = type_account
        self.cursor = cursor

    def run(self):
        return AccountGateway(self.cursor).addAccount(self.login, self.password, self.phone,
self.mail, self.type_account)
```

Класс Authorization для выполнения сценария авторизации:

```
class Authorization(InterfaceRun):
    def __init__(self, login, password, cursor):
        super().__init__()
        self.login = login
```

```

self.password = password

self.cursor = cursor

def run(self):
    acc = AccountGateway(self.cursor).getAccount(self.login)
    if acc == []:
        return False
    if acc[0]["password"] == self.password:
        return True
    else:
        return False

```

Класс AddDiet для выполнения сценария добавления диеты:

```

class AddDiet(InterfaceRun):
    def __init__(self, name, description, age, height, weight, special, cursor):
        super().__init__()
        self.name = name
        self.description = description
        self.age = age
        self.height = height
        self.weight = weight
        self.special = special
        self.cursor = cursor

    def run(self):
        return DietGateway(self.cursor).addDiet(self.name, self.description, self.age, self.height,
self.weight, self.special)

```

Класс FillAboutMe для выполнения сценария заполнения информации о пользователе:

```

class FillAboutMe(InterfaceRun):

```

```

def __init__(self, age, height, weight, special, login, cursor):
    super().__init__()
    self.age = age
    self.height = height
    self.weight = weight
    self.special = special
    self.login = login
    self.cursor = cursor

def run(self):
    return AccountGateway(self.cursor).updateAccount(self.age, self.height, self.weight,
self.special, self.login)

```

Класс FindDiet для выполнения сценария подбора диеты:

```

class FindDiet(InterfaceRun):
    def __init__(self, login, cursor):
        super().__init__()
        self.login = login
        self.cursor = cursor

    def run(self):
        account = AccountGateway(self.cursor).getAccount(self.login)
        if account == []:
            return False, False

        diet = DietGateway(self.cursor).selectDiet(account[0]['age'], account[0]['height'],
                                                    account[0]['weight'], account[0]['special'])

        if diet == []:
            return False, False

        diet_id = diet[0]["id"]
        diet_name = diet[0]["name"]

```



```
HistoryDietGateway(self.cursor).addDiet(account=account[0]['id'], diet=diet[0]["id"])

return diet_name, DishesGateway(self.cursor).getDishes(diet_id)
```

Класс GetHistoryDiet для выполнения сценария получения истории диет:

```
class GetHistoryDiet(InterfaceRun):
    def __init__(self, login, cursor):
        super().__init__()
        self.login = login
        self.cursor = cursor

    def run(self):
        diets = HistoryDietGateway(self.cursor).getHistoryDiet(self.login)
        dishes = {}
        for diet in diets:
            diet_name = diet['name']
            dish_in_query = []
            for name in
DishesGateway(self.cursor).getDishes(DietGateway(self.cursor).getDiet(name=diet_name)[0]['
id']):
                dish_in_query.append(name)
            dishes[diet_name] = dish_in_query
        return diets, dishes
```

Класс RestorePassword – выполнения сценария восстановления пароля:

```
class RestorePassword(InterfaceRun):
    def __init__(self, login, phone, password, cursor):
        super().__init__()
        self.login = login
        self.phone = phone
```

```

self.password = password

self.cursor = cursor

def run(self):
    acc = AccountGateway(self.cursor)
    res = acc.getAccount(self.login)
    if res == []:
        return False
    return acc.updatePasswordAccount(self.login, res[0]['password'])

```

Реализация классов Table Data Gateway

Класс AccountGateway – шлюз таблицы данных для таблицы Account:

```

class AccountGateway:
    def __init__(self, cursor):
        self.cursor = cursor

    def addAccount(self, login, password, phone, mail, type_account):
        try:
            self.cursor.execute("INSERT INTO Account (login, password, phone, mail, type_account)
VALUES (?, ?, ?, ?, ?)",
                                (login, password, phone, mail, type_account))
        except:
            return False
        return True

    def updateAccount(self, age, height, weight, special, login):
        try:
            self.cursor.execute("UPDATE Account SET age = ?, height = ?, weight = ?, special = ?
WHERE login = ?",
                                (age, height, weight, special, login))
        except:

```

```

        return False

    return True

    def updatePasswordAccount(self, password, login):
        try:
            self.cursor.execute("UPDATE Account SET password = ? WHERE login = ?", (password,
login))
        except:
            return False

        return True

    def getAccount(self, login):
        self.cursor.execute(
            "SELECT login, password, phone, mail, type_account, age, height, weight, special, id
FROM Account WHERE login = ?",
            (login,))
        query = self.cursor.fetchall()

        res = []
        for str_db in query:
            res.append({"login": str_db[0], "password": str_db[1], "phone": str_db[2], "mail": str_db[3],
                "type_account": str_db[4], "age": str_db[5], "height": str_db[6], "weight": str_db[7],
                "special": str_db[8], "id": str_db[9]})

        return res

```

Класс DietGateway– шлюз таблицы данных для таблицы Diet:

```

class DietGateway:
    def __init__(self, cursor):
        self.cursor = cursor

    def addDiet(self, name, description, age, height, weight, special):
        try:

```

```

        self.cursor.execute(
            "INSERT INTO Diet (name, description, age, height, weight, special) VALUES (?, ?, ?,
            ?, ?, ?)",
            (name, description, age, height, weight, special))
    except:
        return False
    return True

def updateDiet(self, old_name, name, description, age, height, weight, special):
    try:
        self.cursor.execute(
            "UPDATE Diet SET name = ?, description = ?, age = ?, height = ?, weight = ?, special
            = ? WHERE name = ?",
            (name, description, age, height, weight, special, old_name))
    except:
        return False
    return True

def getDiet(self, name):
    self.cursor.execute("SELECT name, description, age, height, weight, special FROM Diet
    WHERE name = ?", (name,))
    query = self.cursor.fetchall()

    res = []
    for str_db in query:
        res.append({"name": str_db[0], "description": str_db[1], "age": str_db[2], "height":
        str_db[3],
            "weight": str_db[4], "special": str_db[5]})
    return res

def getIdDiet(self, name):
    self.cursor.execute("SELECT id FROM Diet WHERE name = ?", (name,))
    query = self.cursor.fetchall()

```

```

res = []

for str_db in query:
    res.append({"id": str_db[0]})

return res


def getDiets(self):
    self.cursor.execute("SELECT id, name, description, age, height, weight, special FROM Diet")
    query = self.cursor.fetchall()

    res = []
    for str_db in query:
        res.append({"id": str_db[0], "name": str_db[1], "description": str_db[2], "age": str_db[3],
                    "height": str_db[4], "weight": str_db[5], "special": str_db[6]})

    return res


def selectDiet(self, age, height, weight, special):
    self.cursor.execute("SELECT id, name, description, age, height, weight, special FROM Diet
WHERE "

                        "age > ? - 5 AND age < ? + 5 AND height < ? + 5 AND height > ? - 5"

                        " AND weight < ? + 5 AND weight > ? - 5 AND special = ?", (age, age, height,
height, weight, weight, special,))
    query = self.cursor.fetchall()

    res = []
    for str_db in query:
        res.append({"id": str_db[0], "name": str_db[1], "description": str_db[2], "age": str_db[3],
                    "height": str_db[4], "weight": str_db[5], "special": str_db[6]})

    return res


def deleteDiet(self, name, description, age, height, weight, special):
    try:
        self.cursor.execute(

```

```
"DELETE FROM Diet WHERE name = ?, description = ?, age = ?, height = ?, weight =
?, special = ?",
    (name, description, age, height, weight, special))
except:
    return False
return True
```

Класс DishesGateway– шлюз таблицы данных для таблицы Dishes:

```
class DishesGateway:
    def __init__(self, cursor):
        self.cursor = cursor

    def addDish(self, name, diet):
        try:
            self.cursor.execute("INSERT INTO Dishes (name, diet) VALUES (?, ?)", (name, diet))
        except:
            return False
        return True

    def updateDish(self, old_name, name):
        try:
            self.cursor.execute("UPDATE Dishes SET name = ? WHERE name = ?", (name,
old_name))
        except:
            return False
        return True

    def getDish(self, id_dish):
        self.cursor.execute("SELECT name FROM Dishes WHERE id = ?", (id_dish,))
        query = self.cursor.fetchall()

        res = []
```

```

    for str_db in query:
        res.append({"name": str_db[0]})
    return res

def getDishes(self, diet):
    self.cursor.execute("SELECT name FROM Dishes WHERE diet = ?", (diet,))
    query = self.cursor.fetchall()

    res = []
    for str_db in query:
        res.append({"name": str_db[0]})
    return res

def deleteDish(self, name):
    try:
        self.cursor.execute("DELETE FROM Dish WHERE name = ?", (name,))
    except:
        return False
    return True

```

Класс HistoryDietGateway– шлюз таблицы данных для таблицы HistoryDiet:

```

class HistoryDietGateway:
    def __init__(self, cursor):
        self.cursor = cursor

    def addDiet(self, account, diet):
        try:
            self.cursor.execute("INSERT INTO HistoryDiet (account, diet) VALUES (?, ?)", (account, diet))
        except:

```


3.4 Диаграммы последовательностей для иллюстрации работы паттернов

Диаграмма последовательностей для прецедента добавление блюда представлена на рис. 26.

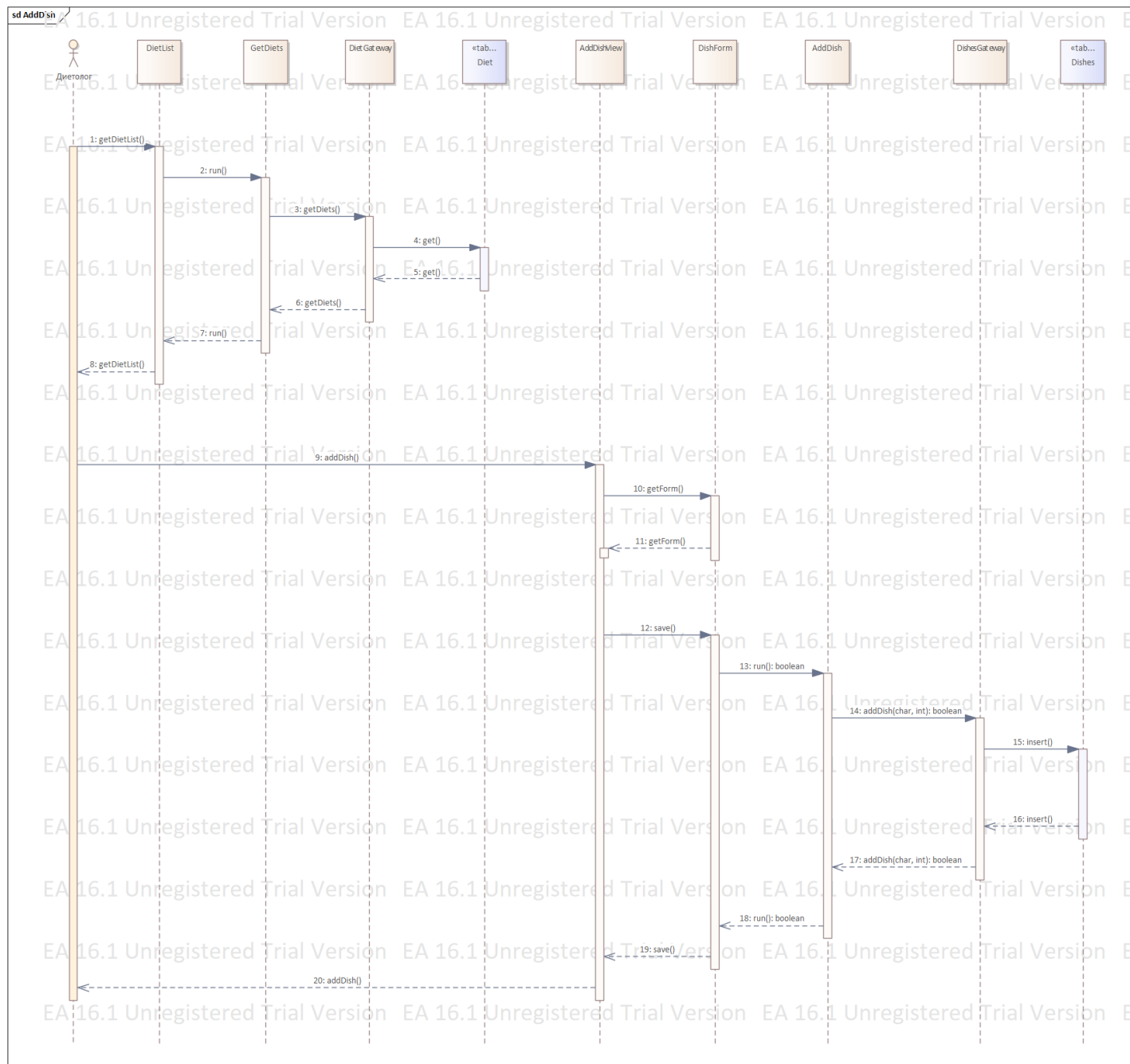


Рисунок 26 - Диаграмма последовательностей для прецедента добавление блюда

Диаграмма последовательностей для прецедента авторизация представлена на рис. 27.

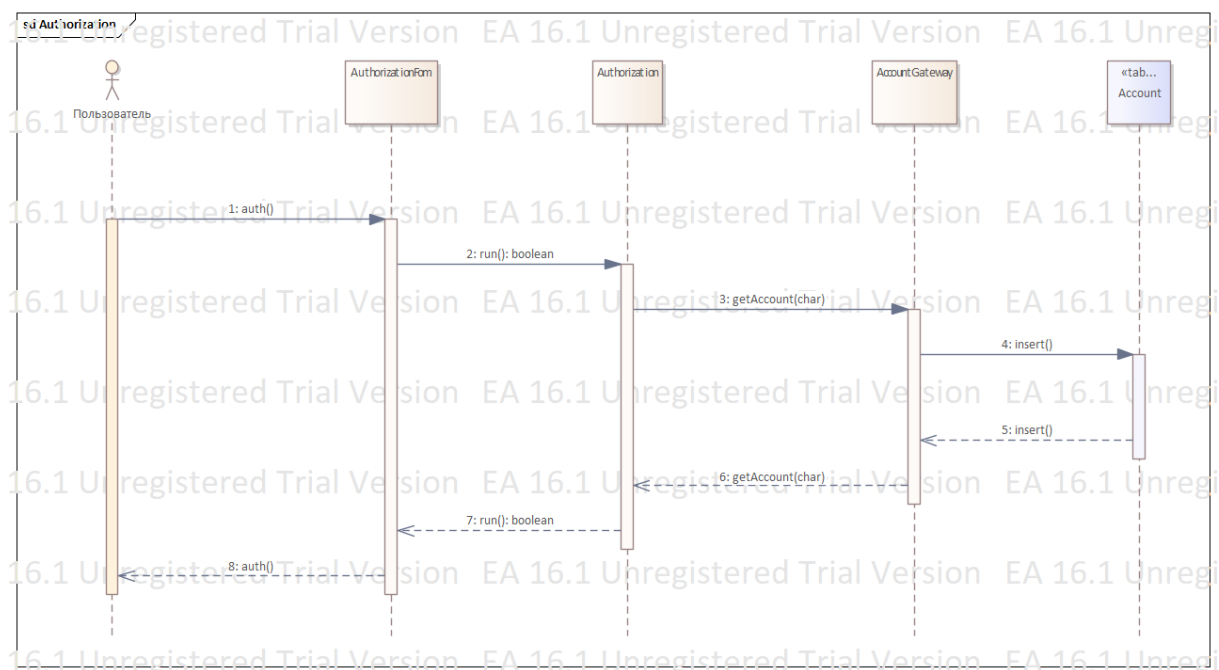


Рисунок 27 - Диаграмма последовательностей для прецедента авторизация

Диаграмма последовательностей для прецедента создание диеты представлена на рис. 28.

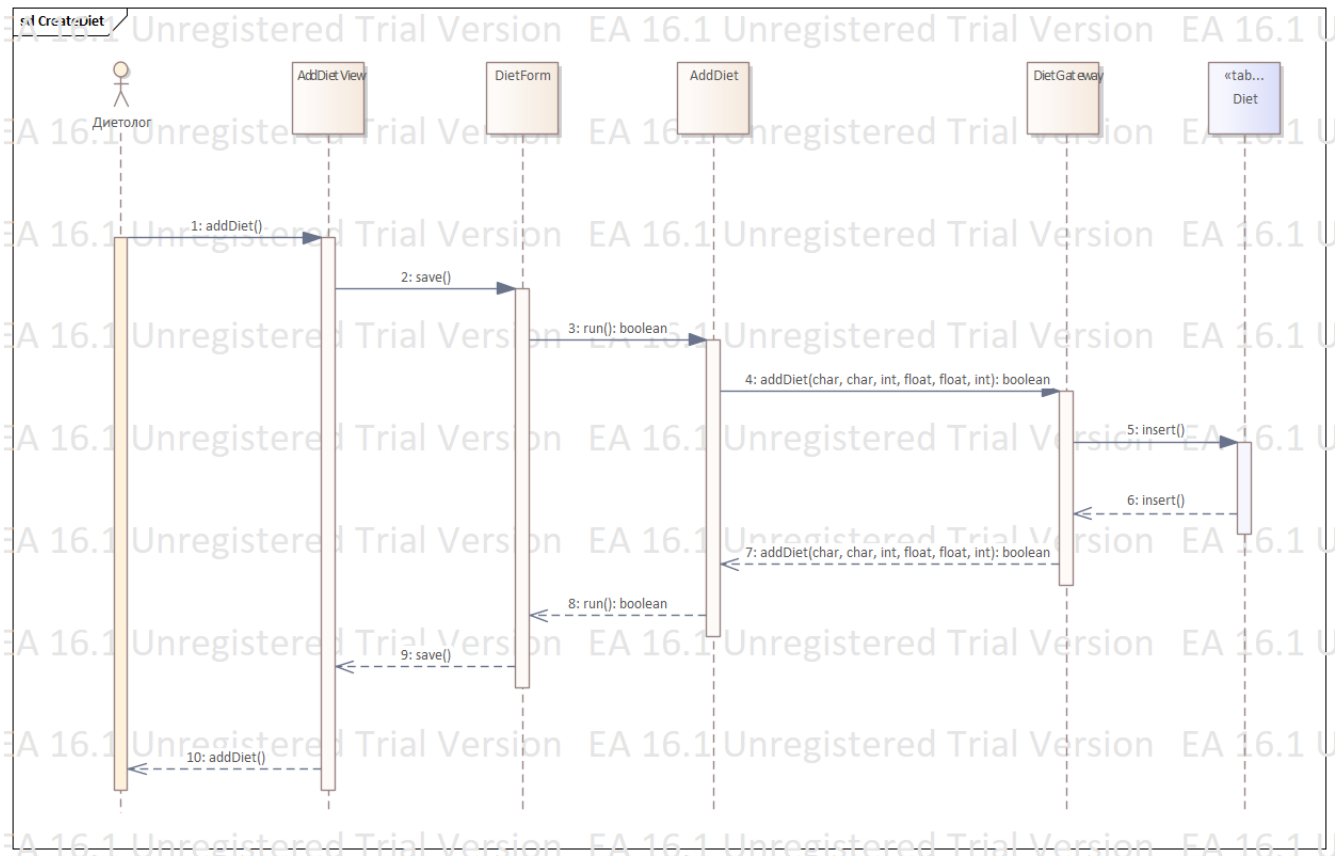


Рисунок 28 - Диаграмма последовательностей для прецедента создание диеты

Диаграмма последовательностей для прецедента заполнение характеристик человека представлена на рис. 29.

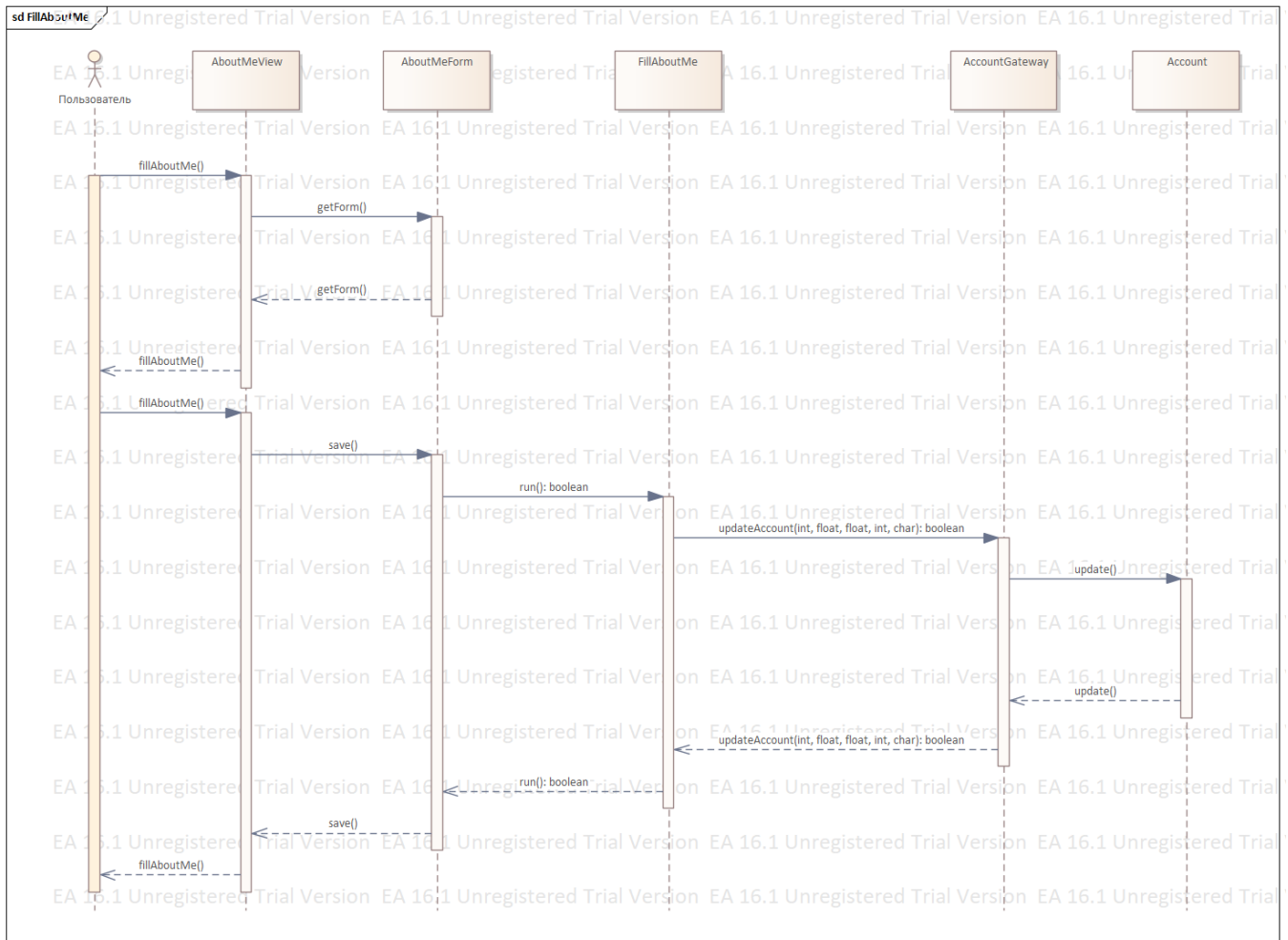


Рисунок 29 - Диаграмма последовательностей для прецедента заполнение данных о себе

Диаграмма последовательностей для прецедента подбор диеты
представлена на рис. 30.

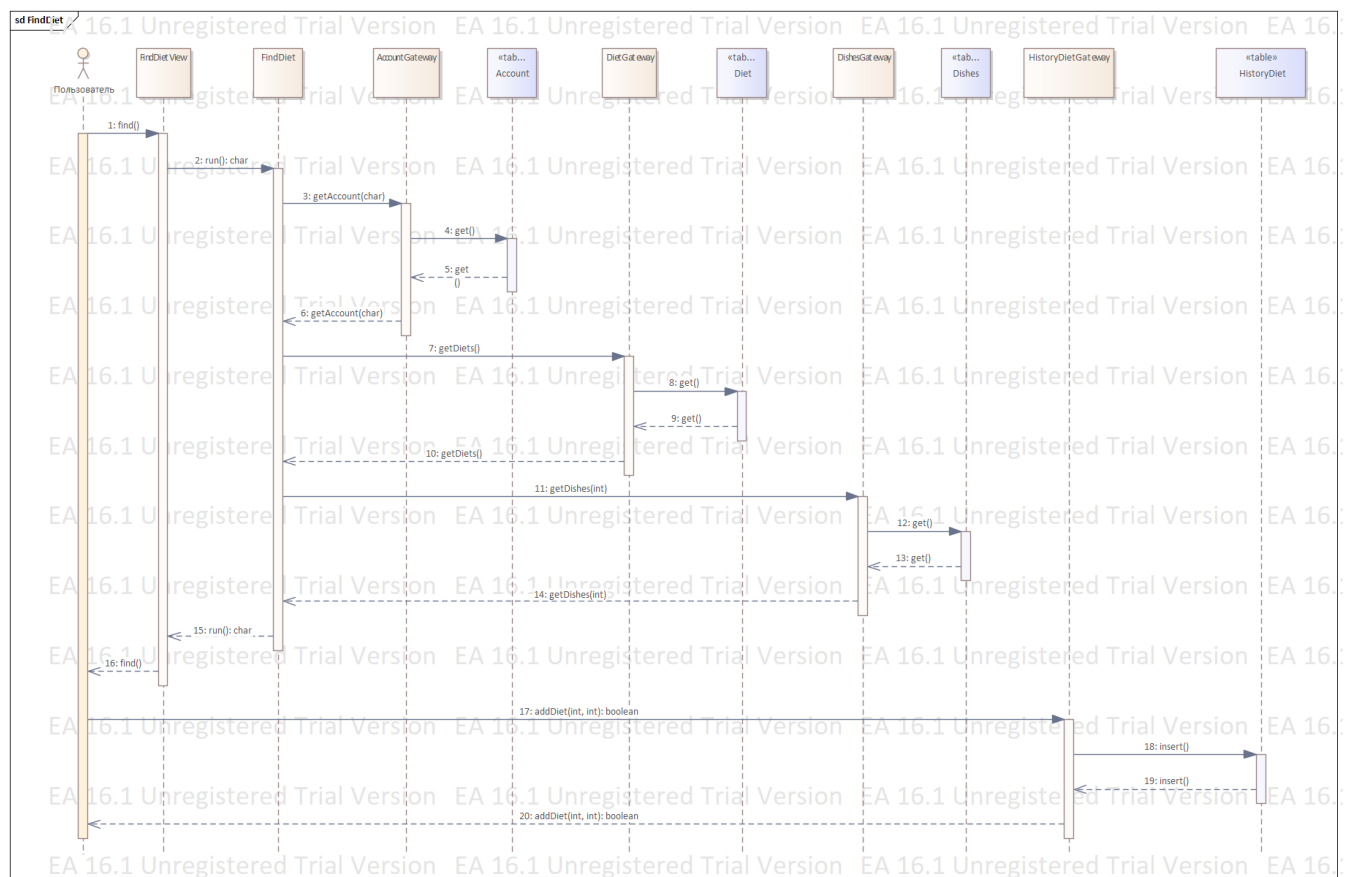


Рисунок 30 - Диаграмма последовательностей для прецедента подбор диеты

Диаграмма последовательностей для прецедента получение истории диет
представлена на рис. 31.

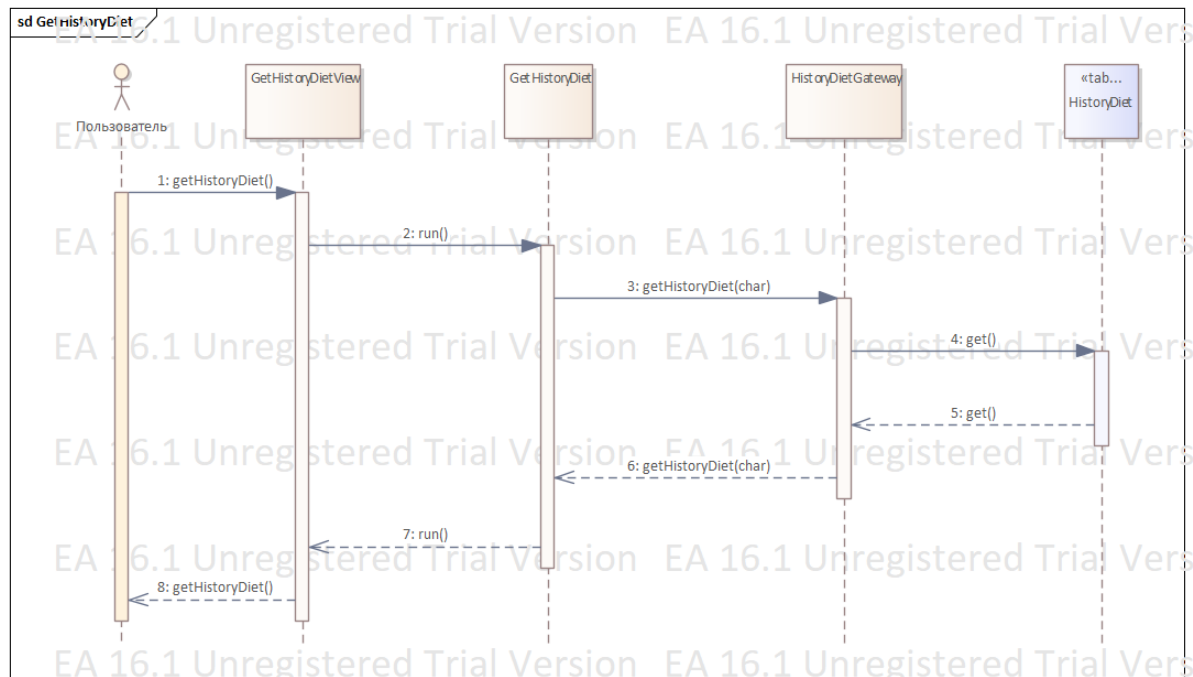


Рисунок 31 - Диаграмма последовательностей для прецедента получение истории диет

Диаграмма последовательностей для прецедента регистрация представлена на рис. 32.

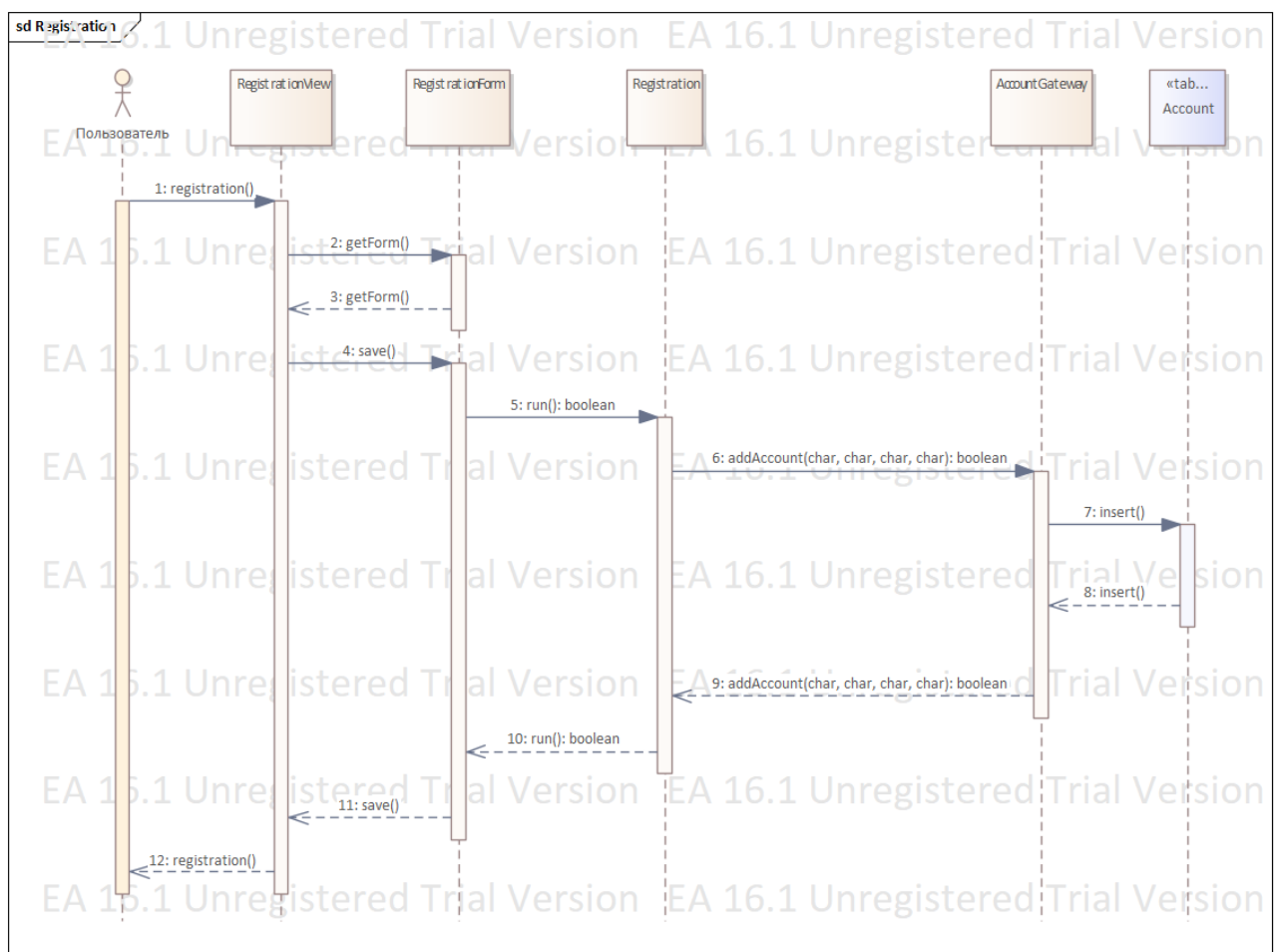


Рисунок 32 - Диаграмма последовательностей для прецедента регистрация

Диаграмма последовательностей для прецедента восстановление пароля представлена на рис. 33.

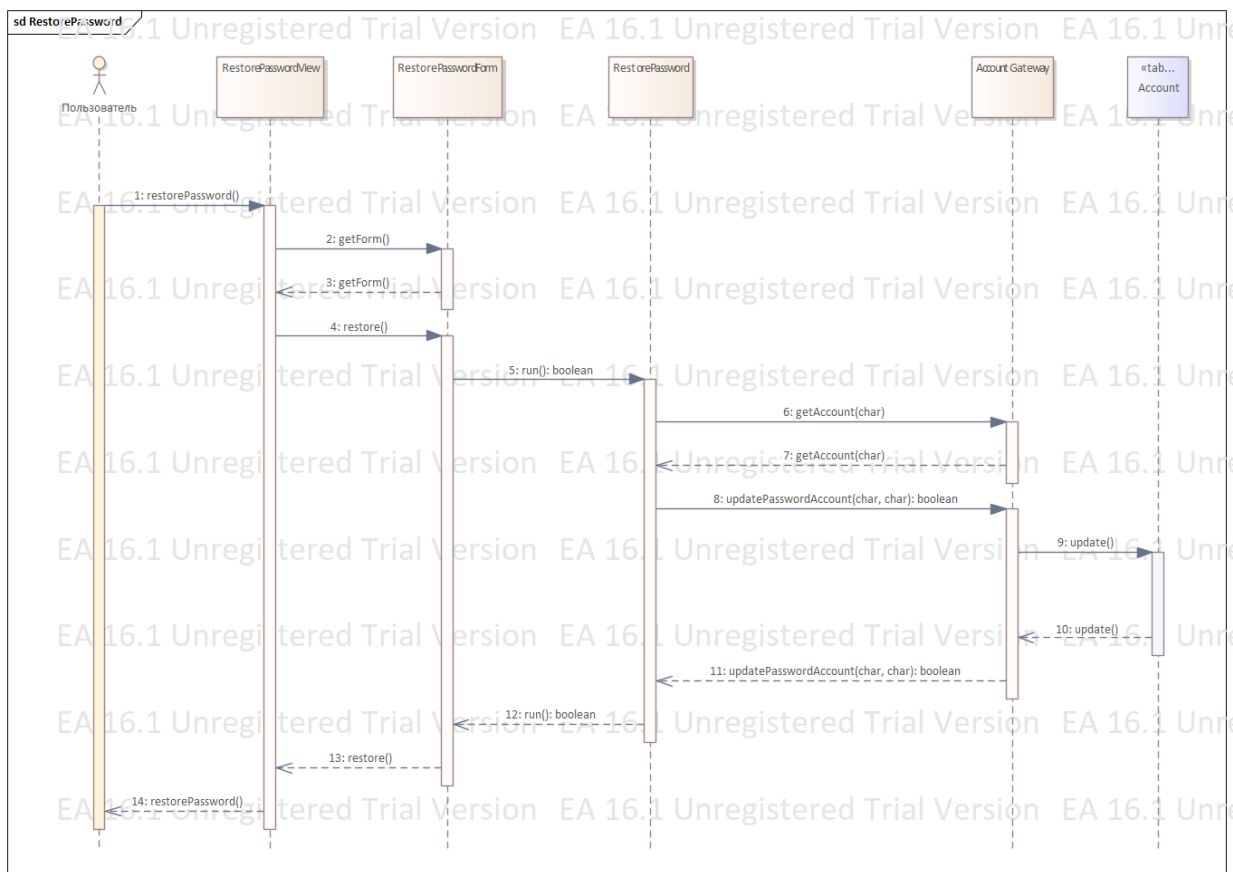


Рисунок 33 - Диаграмма последовательностей для прецедента восстановление пароля

3.5 Перечень и последовательность проведения тестов

Проведение функционального тестирования проводится для прецедента “Подбор диеты”.

- Пользователь должен нажать кнопку “Подобрать диету” в главном меню.

Входные параметры:

- Login. Логин пользователя, представляющий из себя строку типа str.
- Age. Возраст, представляющий из себя число типа int.
- Height. Рост, представляющий из себя число типа float.
- Weight. Вес, представляющий из себя число типа float.
- Special. Особенности, представляющий из себя строку типа str.

Содержание тестовых таблиц в базе данных, представлены в табл. 9 и табл. 10 для диеты и блюд соответственно.

Таблица 9 – Содержание тестовой таблицы diet

id	name	description	age	height	weight	special
1	Название диеты с блюдом	Описание	18	180	80	Здоровый
2	Название диеты без блюд	Описание	19	190	90	Здоровый

Таблица 10 – Содержание тестовой таблицы dishes

id	name	diet
1	Название блюда	1

Функциональное тестирование прецедента подбора диеты, представлено на табл. 11.

Таблица 11 - Функциональное тестирование прецедента подбора диеты

Тест кейс	Входные данные	Выходные данные
Корректные запрос	Login = “user_login” Age = 18 Height = 180 Weight = 80 Special = “Здоровый”	Диета: Название диеты с блюдом Блюда: Название блюда
Не найдена подходящая диета	Login = “user_login” Age = 1 Height = 300 Weight = 10 Special = “Здоровый”	Диета: Не найдена

Продолжение таблицы 11

У найденной диеты отсутствуют блюда	Login = "user_login" Age = 19 Height = 190 Weight = 90 Special = "Здоровый"	Диета: Название диеты без блюد Блюда: Не содержит
--	---	---

Тесты модулей происходят в следующей последовательности:

1. Модуль с классами, которые созданы согласно паттерну Transaction script
2. AccountGateway
3. DietGateway
4. DishGateway
5. HistoryDietGateway

Были протестированы все gateway-классы и классы транзакции. Тесты реализованы с помощью библиотеки pytestcoverage с использованием команды:

```
coverage run -m unittest classes.py, tests\Run_test.py,
tests\AccountGateway_test.py, tests\DietGateway_test.py, tests\DishGateway_test.py,
tests\HistoryDietGateway_test.py.
```

Тесты для каждого метода строятся по следующему принципу:

- Производится мокирование курсор для связи с БД
- Собираются данные для тестирования функции
- Данные передаются в функции
- Результат функции сравнивается с нужным результатом

Пример тестирования получения информации об аккаунте приведён на рис. 34. Этот тест проверяет правильность получения данных об аккаунте.

```
def test_getAccount(self):
    res = [{"login": LOGIN, "password": PASSWORD, "phone": PHONE, "mail": MAIL,
            "type_account": 0, "age": 18, "height": 180, "weight": 80,
            "special": 0, "id": 0}]
    cursor = MagicMock()
    cursor.fetchall = Mock(return_value=[[LOGIN, PASSWORD, PHONE, MAIL, 0, 18, 180,
    acc = AccountGateway(cursor)
    self.assertEqual(first=acc.getAccount(LOGIN), second=res)
```

Рисунок 34 – Тестирование получения информации об аккаунте

Входные данные следующие: "login", "password", "89165554498", "mail@mail.com", "type_account": 0, "age": 18, "height": 180, "weight": 80, "special": 0, "id": 0. Результат тестирования примера, представлен на рис. 35.

```
AccountGateway_test.py::AccountGatewayTest.test_getAccount
Testing started at 21:41 ...
Launching pytest with arguments AccountGateway_test.py::AccountGatewayTest::test_getAccount --n

===== test session starts =====
collecting ... collected 1 item

AccountGateway_test.py::AccountGatewayTest::test_getAccount PASSED [100%]

===== 1 passed in 0.09s =====

Process finished with exit code 0
```

Рисунок 35 – Результат тестирования

Рассмотрим покрытие проекта тестами с помощью той же библиотеки с помощью команды `python -m coverage report`. Получим с ее помощью статистику о покрытии кода 94%. Более точные данные о покрытии представлены на рис. 36.

```

PS C:\Users\A1i5k\PycharmProjects\kp> python -m coverage report
Name                               Stmts  Miss  Cover
-----
classes.py                         264    24    91%
tests\AccountGateway_test.py       48     1    98%
tests\DietGateway_test.py          53     1    98%
tests\DishGateway_test.py          53     1    98%
tests\HistoryDietGateway_test.py    29     1    97%
tests\Run_test.py                   72     1    99%
-----
TOTAL                               519    29    94%

```

Рисунок 36 – Покрытия кода тестами

Глава 4. Этап внедрения

На этапе внедрения происходит активное внедрение разработанного программного продукта в реальную среду. Это включает в себя:

- Применить ПО в среде заказчика.
- Завершить реализацию продукта.

4.1 Перечень программ и рекомендации по установке

Перечень программ разработанной системы:

- 1) Приложение – это исполнимый файл формата .exe для Windows.

Рекомендации по установке:

- 1) Характеристики сервера:
 - a. Интернет: подключение к интернету;
 - b. ОС: Windows 7+;
 - c. ОЗУ: >256 МБ;
 - d. HDD: 512-1024 МБ на жестком диске
- 2) Необходимо установить следующие библиотеки:
 - a. Библиотека DJANGO
 - b. Библиотека SQLite3
- 3) На клиенте необходимо иметь браузер Internet Explorer 1 и выше (или аналогичные)
- 4) Формы страниц
 - a. Для авторизации
 - b. Для регистрации
 - c. Для подбора диеты
 - d. Для основной страницы пользователя
 - e. Для основной страницы диетолога
 - f. Для восстановления пароля
 - g. Для добавления диеты
 - h. Для добавления блюда

4.2 Перечень документации для пользователей и заказчиков

Перечень документации для заказчика:

- Техническое задание
- Программа и методика испытаний

Перечень документации для пользователя:

- Описание системы
- Руководство по установке и эксплуатации системы
- Руководство пользователя для пользователя
- Руководство пользователя для диетолога

4.3 Рекомендации по внедрению

В целях ознакомления с функционалом приложения и способами его использования будут проведены очные семинары. Привлечь больше диетологов для добавления большего количества диет.

Заключение

В данной курсовой работе была реализована система подбора диеты. Система способна подбирать диету в зависимости от заданных пользователем данных о себе, которые так же можно обновить в любой момент. Так же предусмотрена роль аккаунта - диетолога, которая позволяет, добавлять диеты и блюда. Уже завершённые диеты можно посмотреть. Выбор аккаунта производится при регистрации самим регистрирующимся.