|  |  |
| --- | --- |
| lu135925on3bu_tmp_3360867a00ce4d37 | **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 и искусственный интеллект

КАФЕДРА                  Системы обработки информации и управления

**Домашнее задание №2**

### По курсу

### «Технологии разработки программного обеспечения»

**«Применение паттернов проектирования»**

Подготовил:

Студент группы

**ИУ5-14Б Журавлев Н.В**

19.11.2023

Проверила:

**Виноградова М.В.**

*2023 г*.

**Цель работы:**

* Изучить основные паттерны проектирования, их особенности и область применения;
* Получить практические навыки программирования паттернов;
* Освоить технологию включения паттернов в собственную программу.

**Полученное задание:**

1. Разработать программу (основные прецеденты) на основе темы, выданной преподавателем (по варианту).
2. Реализовать в программе паттерны (по варианту) бизнес-логики и работы с БД.
3. Составить набор диаграмм классов и последовательностей, которые демонстрируют структуру и поведение программы.
4. Отдельно составить диаграммы классов и последовательностей для иллюстрации примененных паттернов.

Рассмотрим реализацию паттернов в приложении «Информационная экспертная система по подбору диеты».

Вариант задания:

* Паттерн бизнес логики – transaction script;
* Паттерн работы с БД – Table Data Gateway.

**Ход работы:**

**Выделение классов Table Data Gateway**

Для работы с паттерном Table Data Gateway для каждой таблицы базы данных необходимо создать класс шлюза таблицы. В базе данных разрабатываемого приложения имеется четыре таблицы: Account, Diet, Dishes и HistoryDiet. В соответствии с выбранным паттерном создаем для каждой таблицы по классу шлюзу таблицы. Таким образом, выделяем классы AccountGateway, DietGateway, DishesGateway и HistoryDietGateway.

Класс Account – информация о пользователе. Атрибуты: логин, пароль, id, почта, телефон, тип аккаунта, возраст, вес, рост, особенности.

Методы:

* addAccount(login, password, phone, mail) – добавить новый аккаунт;
* updateAccount(age, height, weight, special, login) – обновить данные о человеке;
* updatePasswordAccount(password, login) – обновить пароль пользователя;
* getAccount(login) – получить данные об аккаунте.

Класс Diet – информация о диете. Атрибуты: возраст, название, описание, id, рост, вес, особенности.

Методы:

* addDiet(name, description, age, height, weight, special) – добавить новую диету;
* updateDiet(name, description, age, height, weight, special, id\_diet) – обновить данные о диете;
* getDiet(name) – получить данные о диете;
* getDiets() – Получить все диеты;
* deleteDiet(name, description, age, height, weight, special) – удалить диету.

Класс Dishes – информация о блюдах. Атрибуты: название, диета, id.

Методы:

* addDish(name, diet) – добавить блюдо;
* updateDish(name, id\_dish) – обновить блюдо;
* getDish(id\_dish) – получить информацию о блюде;
* getDishes(diet) – получить все блюда для выбранной диеты;
* deleteDish(name) – удалить блюдо.

Класс HistoryDiet – история диет пользователя. Атрибуты: id пользователя, id диеты.

Методы:

* addDiet(account, diet) – добавить диету в историю;
* getHistoryDiet(login) – получить историю диет.

**Выделение классов Transaction script**

Способ разнесения кода сценариев транзакции по классам связан с разработкой собственного класса для каждого сценария транзакции: определяется тип, базовый по отношению ко всем командам, в котором предусматривается некий метод выполнения (run), удовлетворяющий логике сценария транзакции. Производные классы переопределяют метод run. Параметры, которыми инициализируются классы сценариев транзакции, передаются в метод run.

Класс Registration – выполнение сценария регистрации. Атрибуты: логин, пароль, почта, телефон.

Методы:

* run() - выполняет регистрацию нового пользователя.

Класс Authorization – выполнение сценария авторизации. Атрибуты: логин, пароль.

Методы:

* run() - выполняет авторизацию.

Класс AddDiet – выполнение сценария добавления диеты. Атрибуты: возраст, название, описание, рост, вес, особенности.

Методы:

* run() - добавляет новую диету.

Класс FillAboutMe – выполнение сценария заполнения информации о пользователе. Атрибуты: возраст, рост, вес, особенности, логин.

Методы:

* run() – заполняет данные о пользователе.

Класс FindDiet – выполнение сценария подбора диеты. Атрибуты: логин.

Методы:

* run() - подбор диеты для пользователя.

Класс GetHistoryDiet – выполнение сценария получения истории диет. Атрибуты: логин.

Методы:

* run() - получение истории диет.

Класс RestorePassword – выполнение сценария восстановления пароля. Атрибуты: логин, новый пароль, телефон.

Методы:

* run() - восстановление пароля.

Класс AddDish – выполнение сценария добавления блюда. Атрибуты: название, id диеты.

Методы:

* run() - добавления блюда к диете.

**Диаграммы классов и последовательностей**

Диаграмма классов для прецедента добавление блюда:

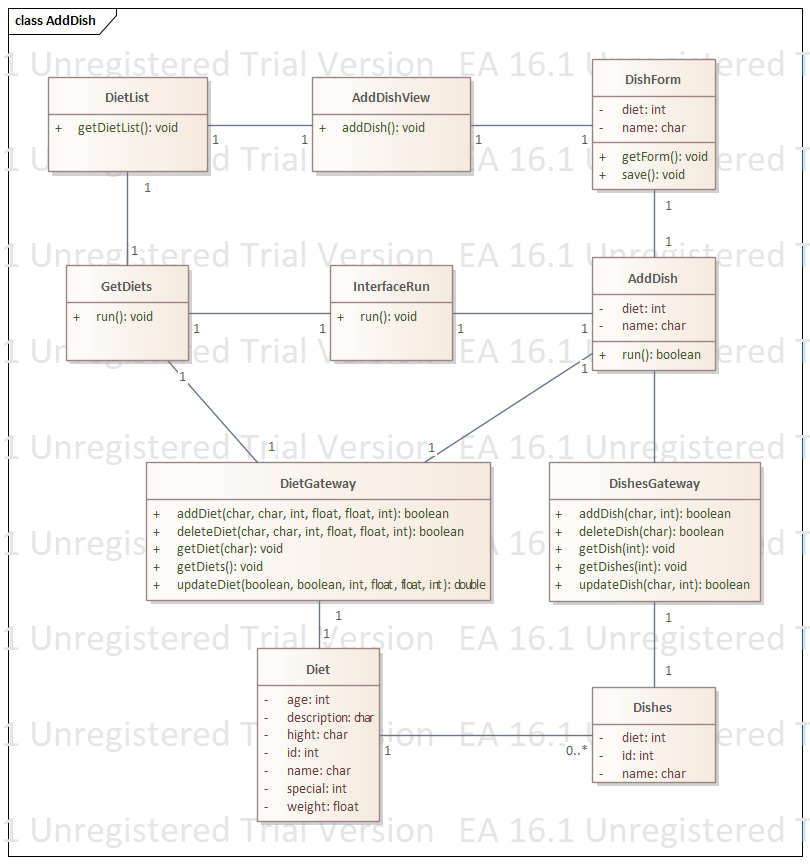


Диаграмма последовательностей для прецедента добавление блюда:

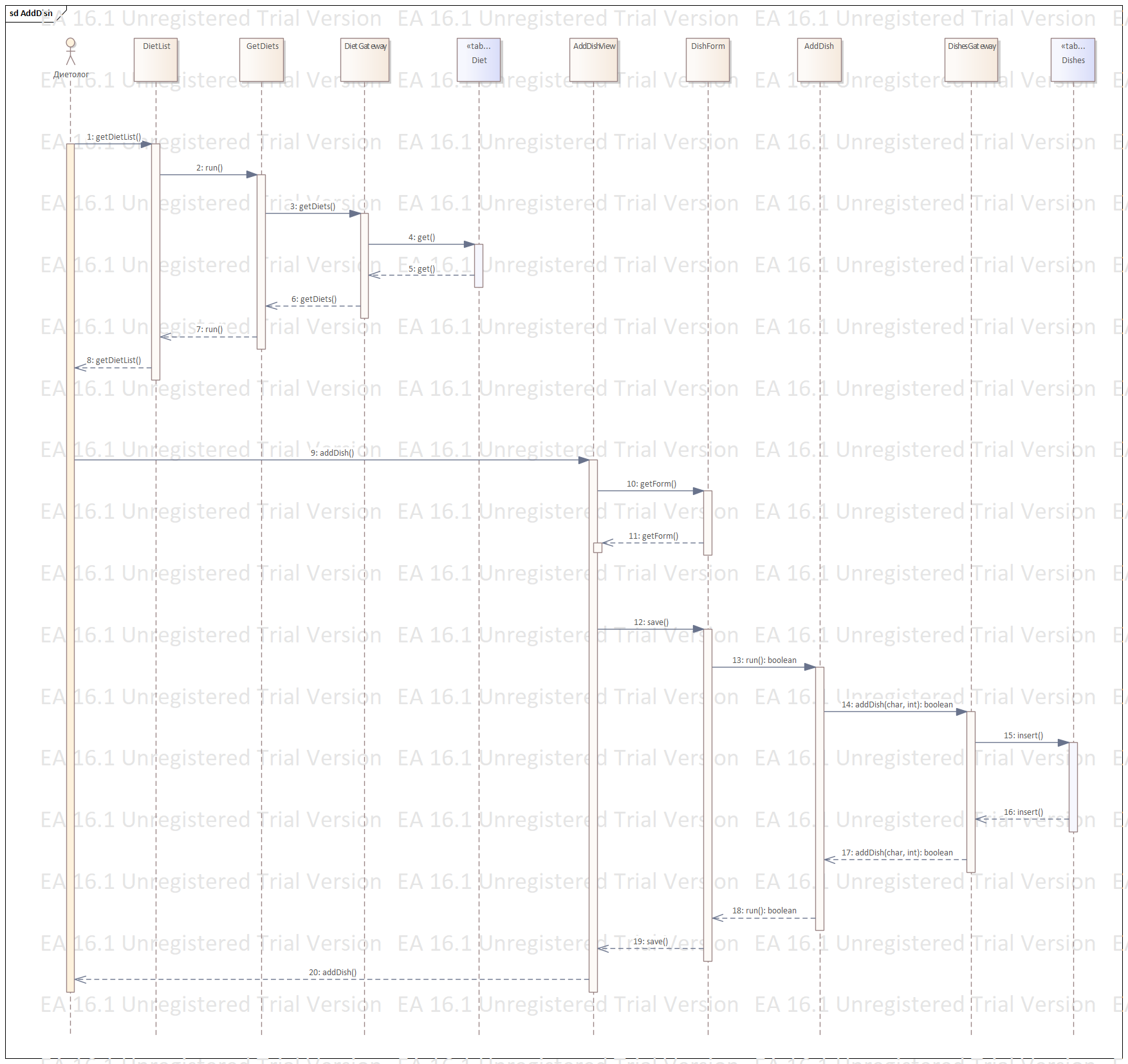


Диаграмма классов для прецедента авторизация:

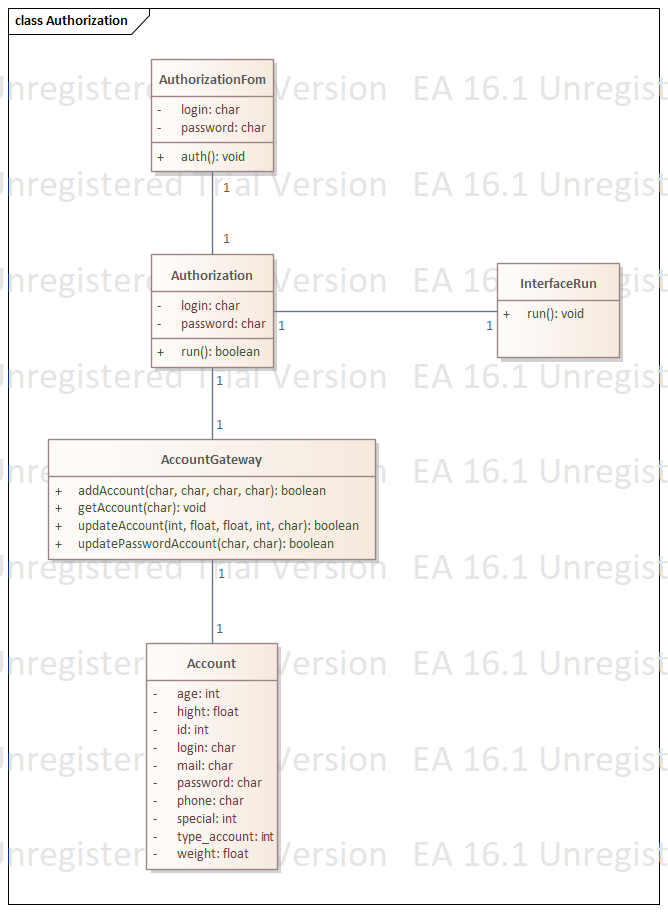


Диаграмма последовательностей для прецедента авторизация:

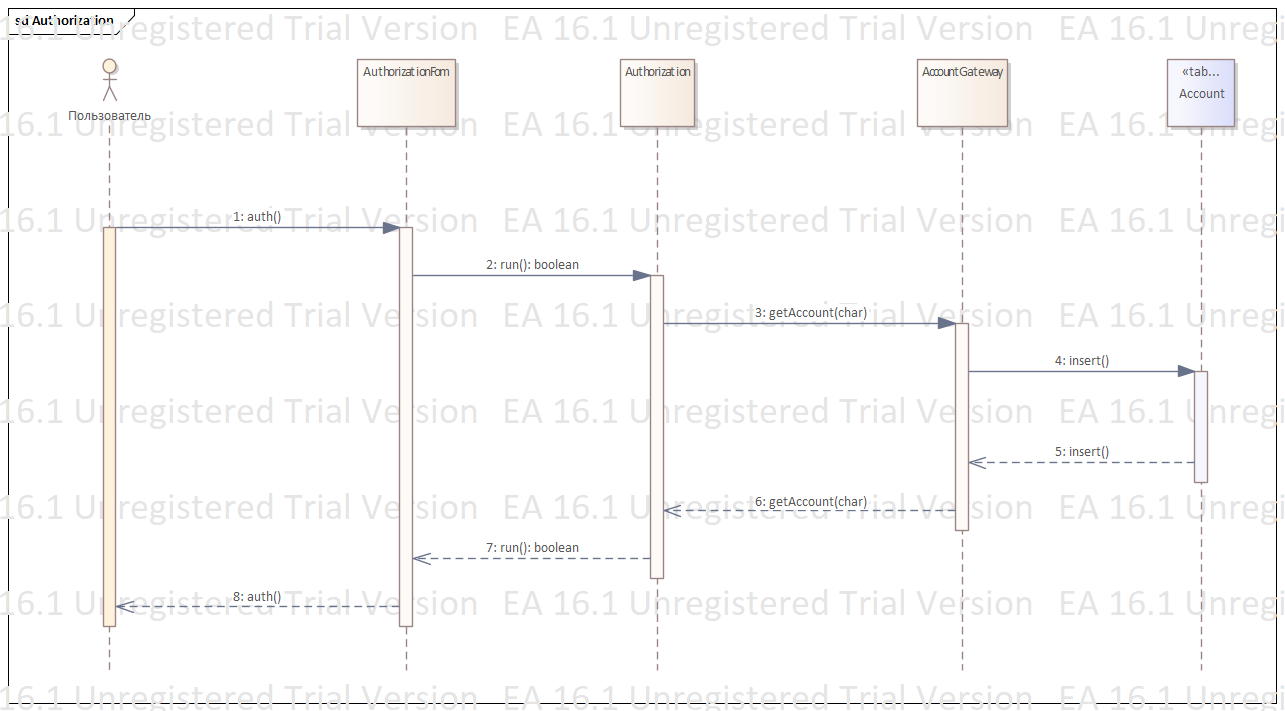


Диаграмма классов для прецедента создание диеты:

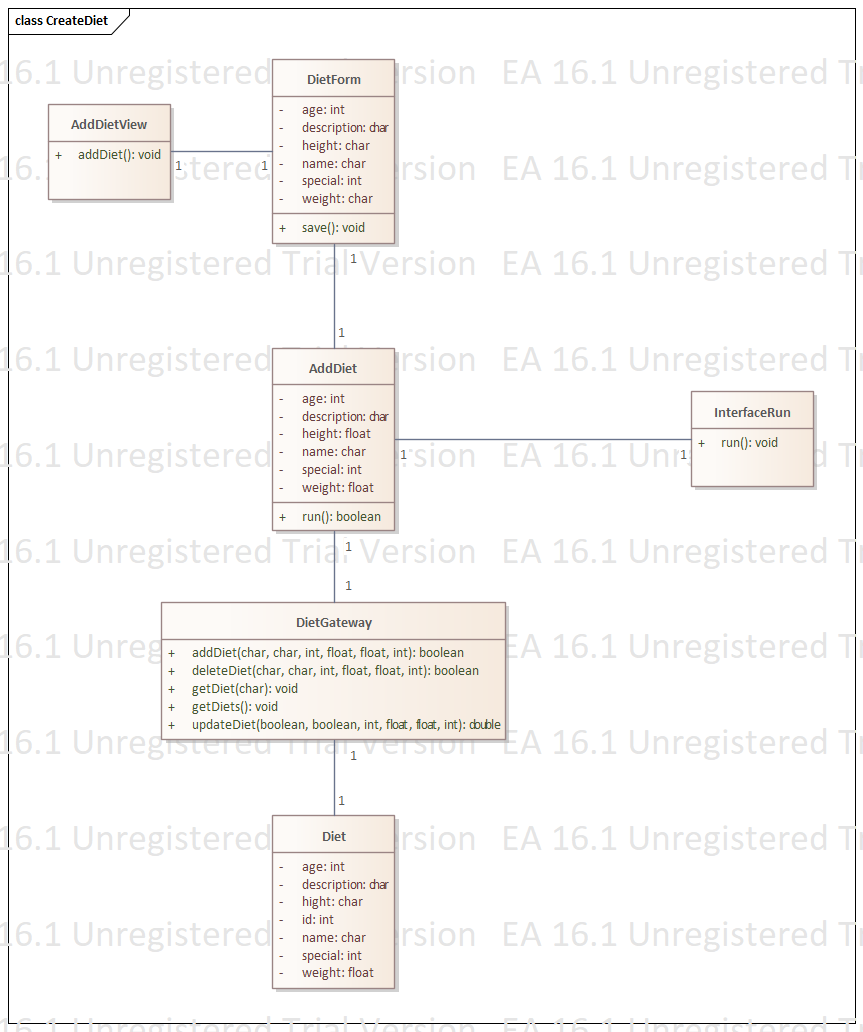


Диаграмма последовательностей для прецедента создание диеты:

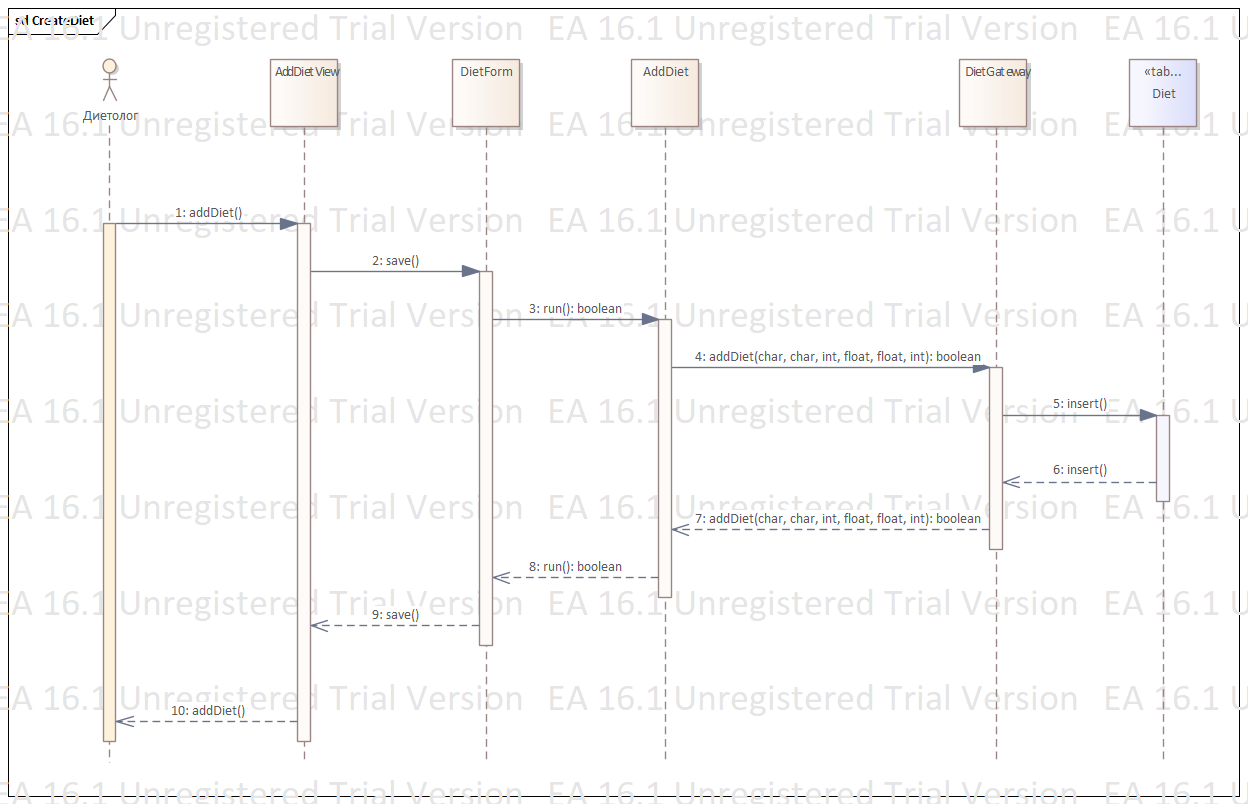


Диаграмма классов для прецедента заполнение характеристик человека:

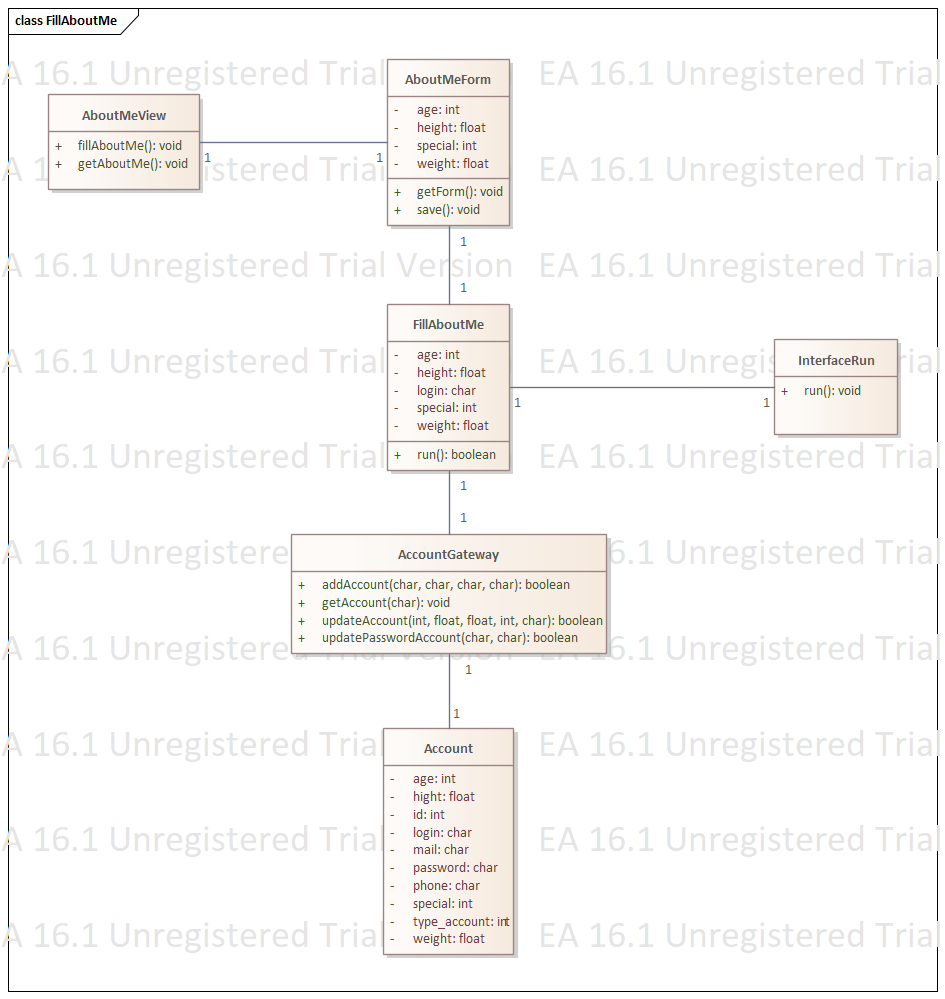


Диаграмма последовательностей для прецедента заполнение характеристик человека:

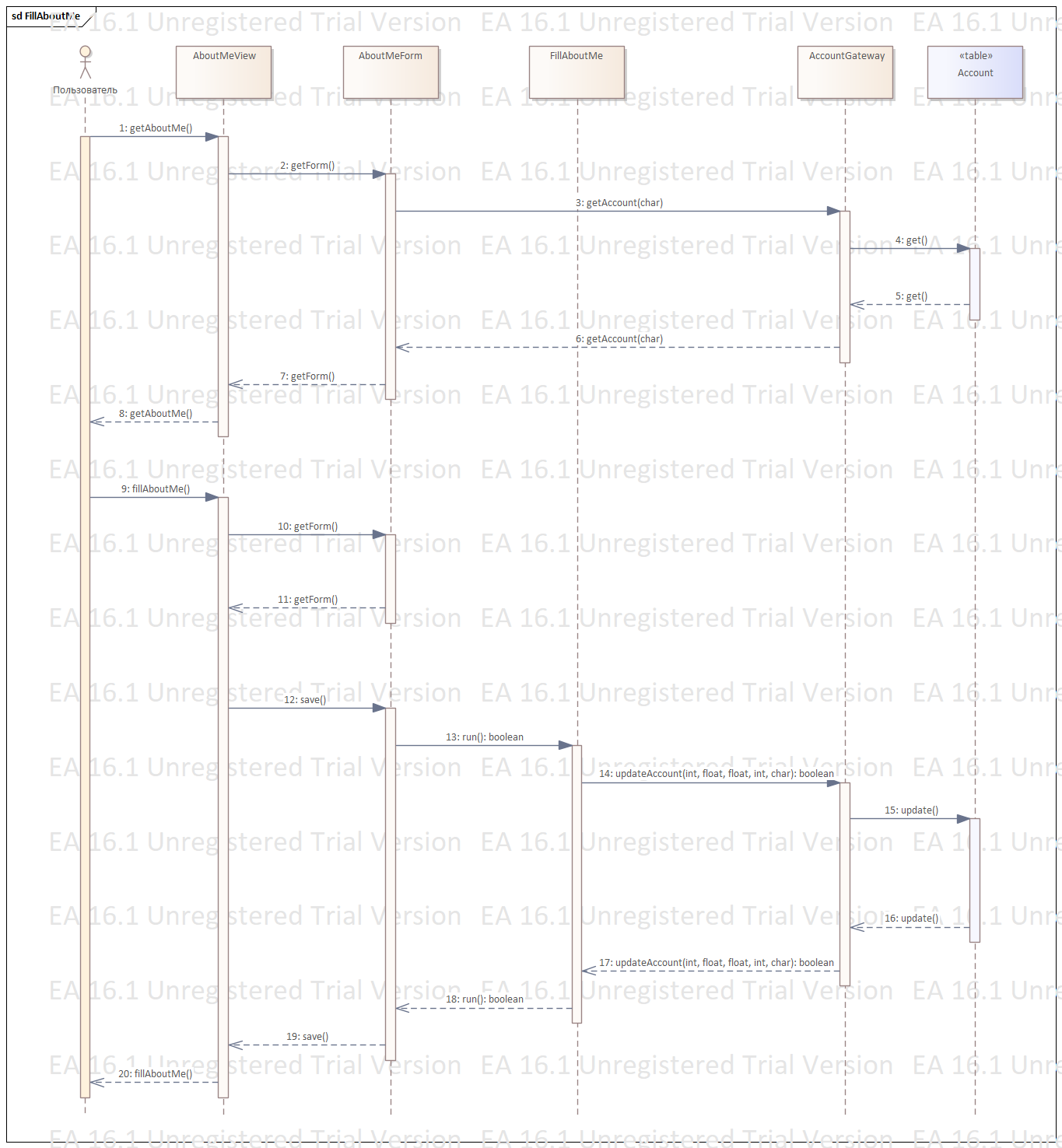


Диаграмма классов для прецедента подбор диеты:

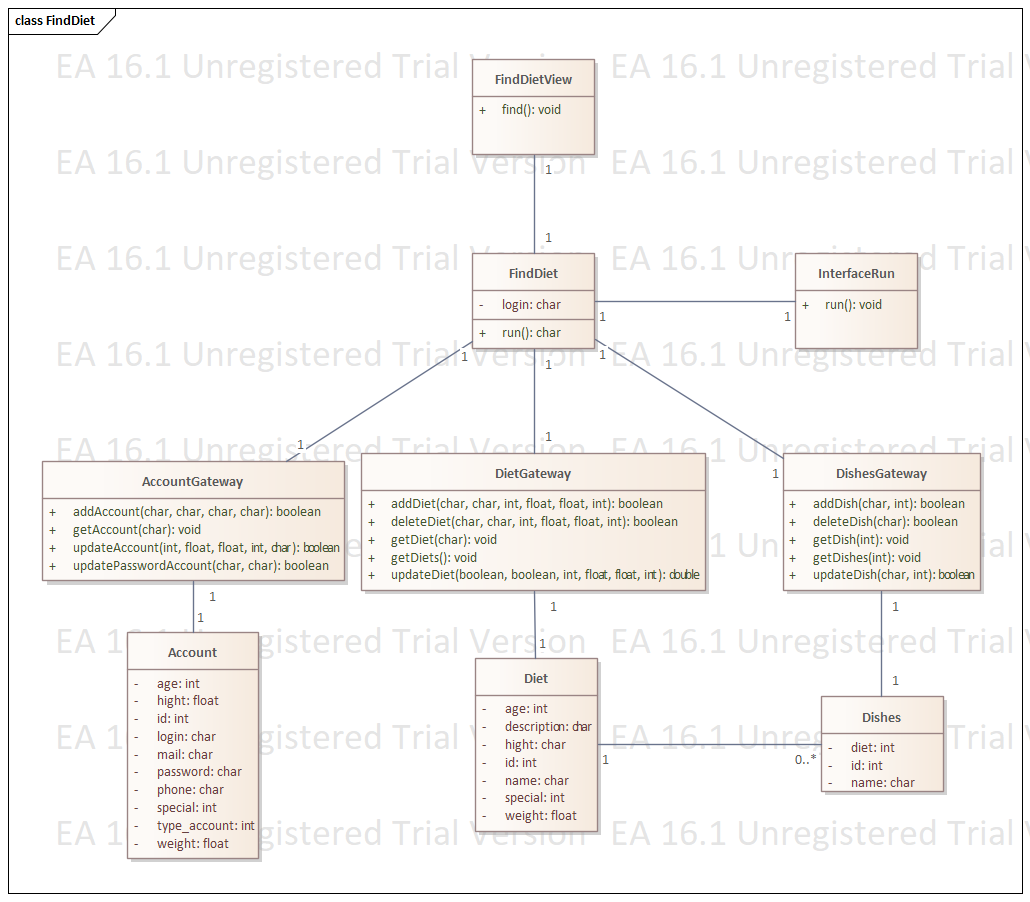


Диаграмма последовательностей для прецедента подбор диеты:

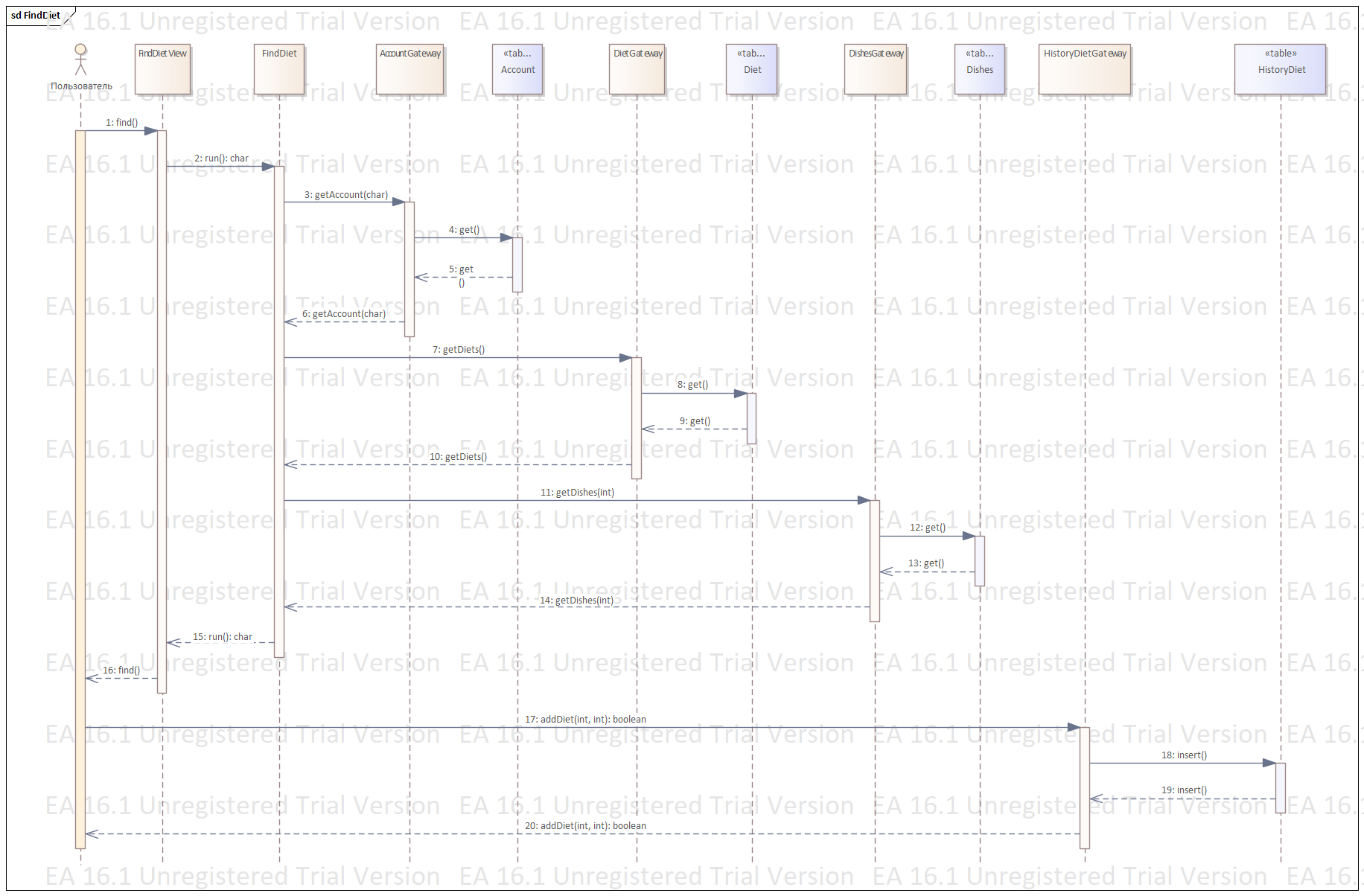


Диаграмма классов для прецедента получение истории диет:

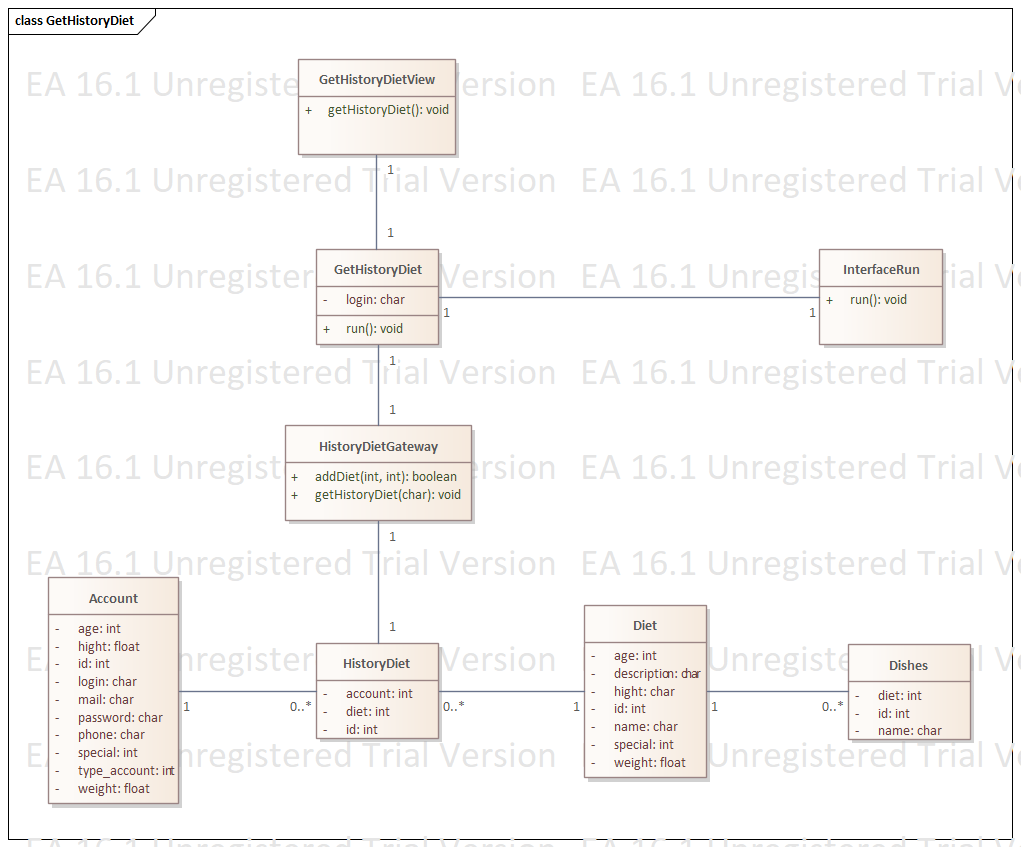


Диаграмма последовательностей для прецедента получение истории диет:

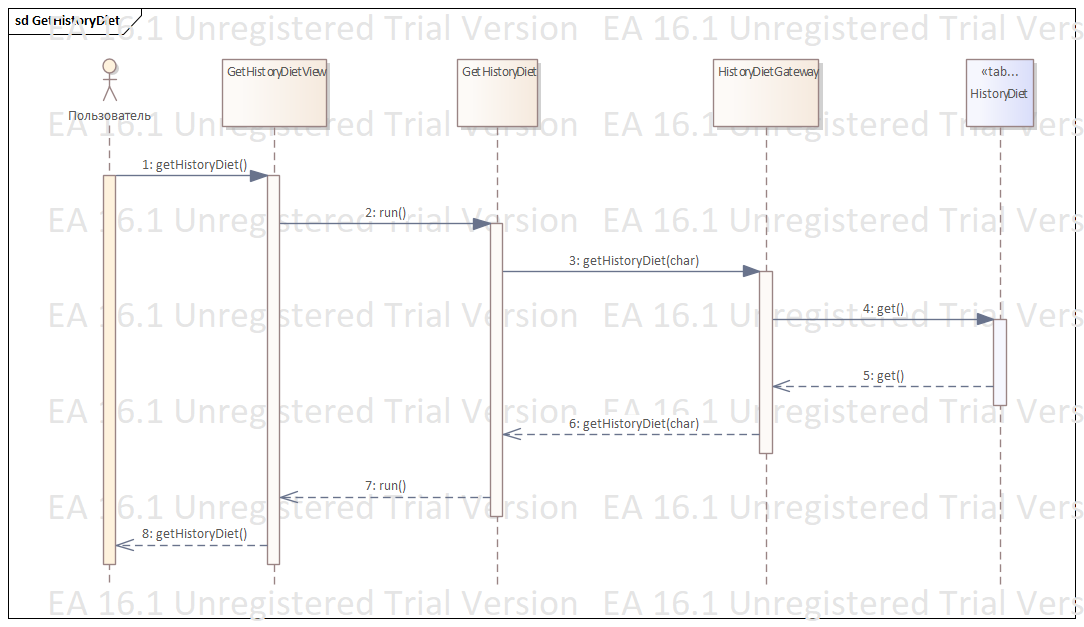


Диаграмма классов для прецедента регистрация:

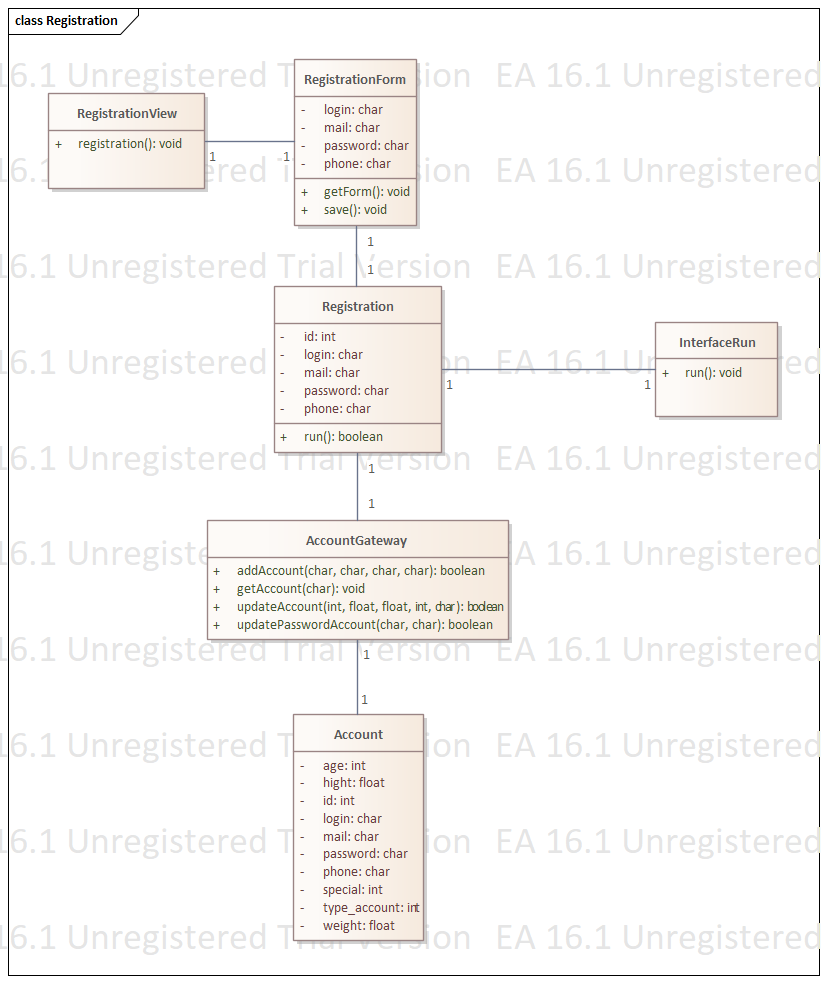


Диаграмма последовательностей для прецедента регистрация:

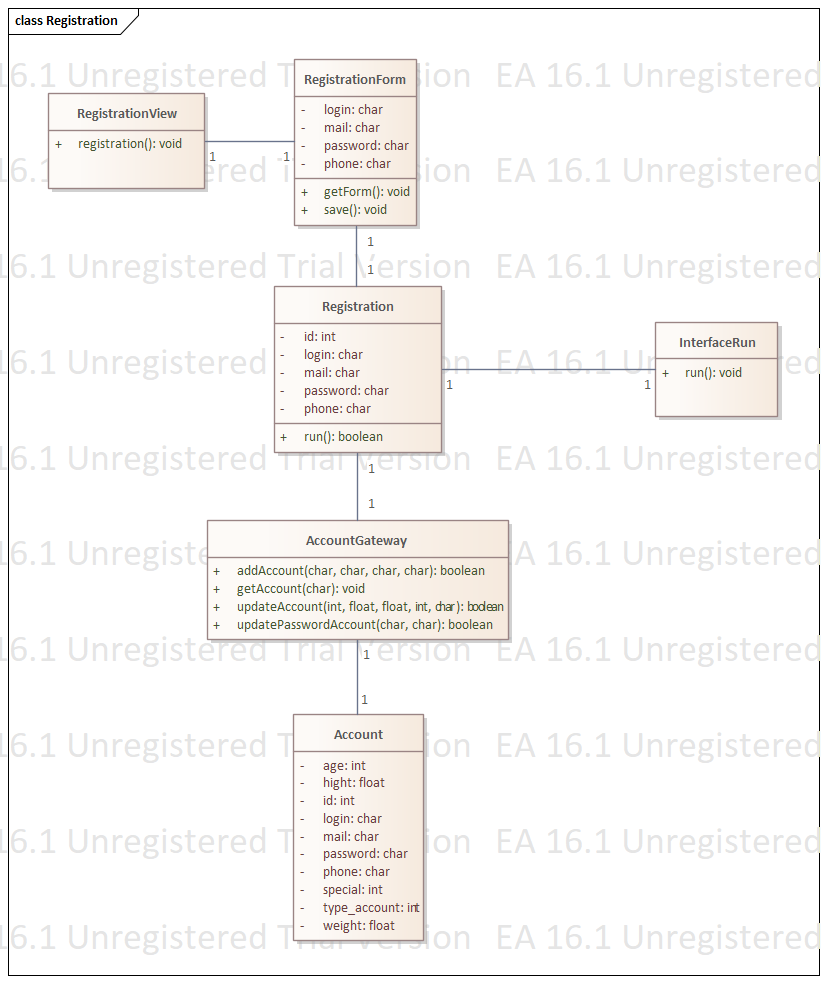
****

Диаграмма классов для прецедента восстановление пароля:

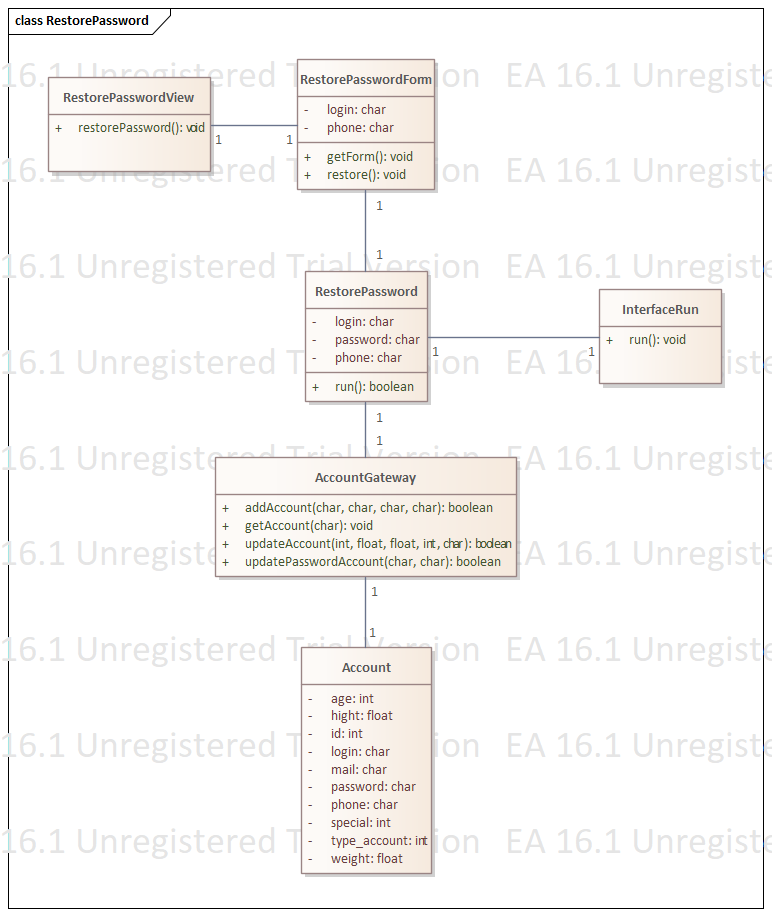
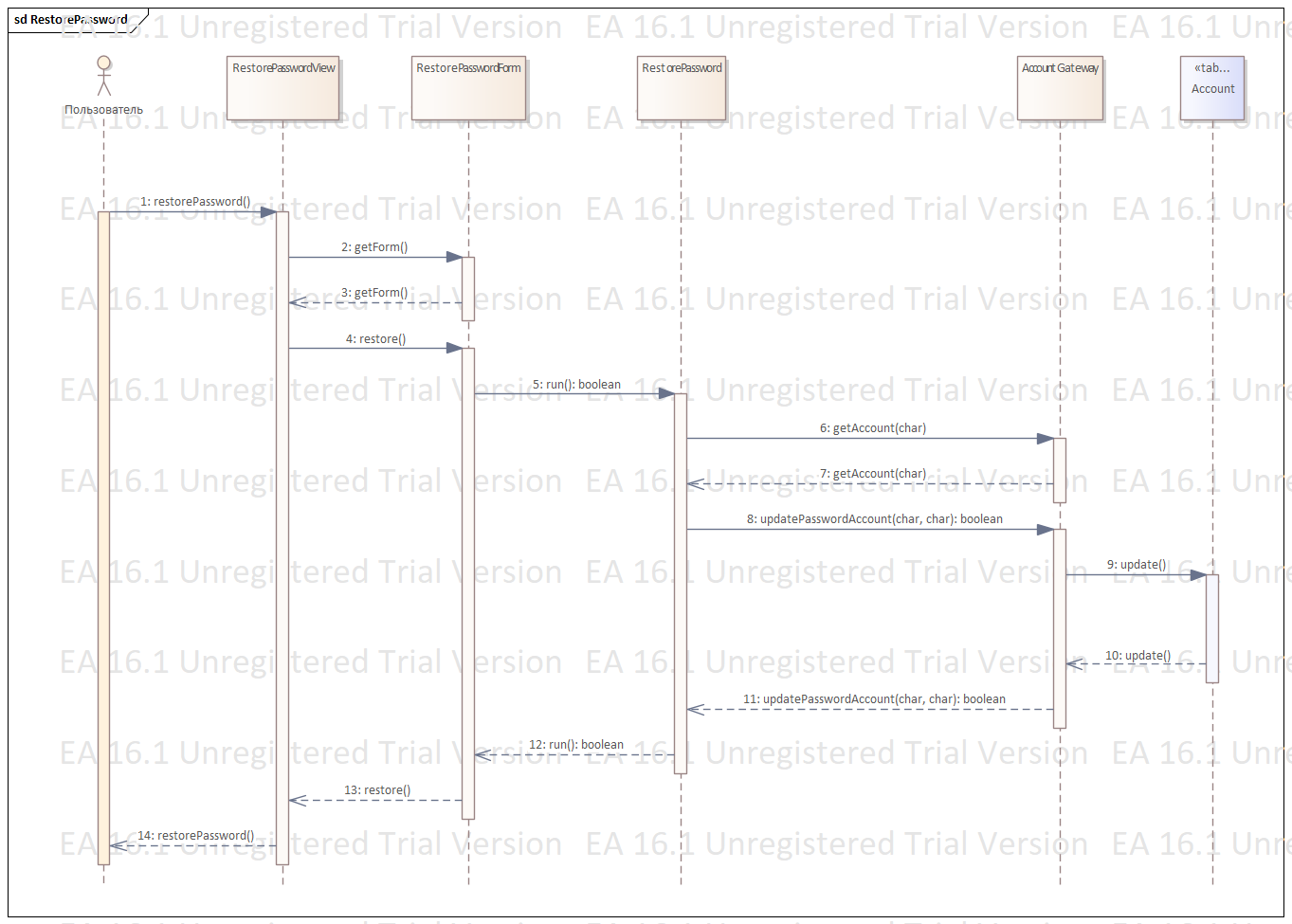
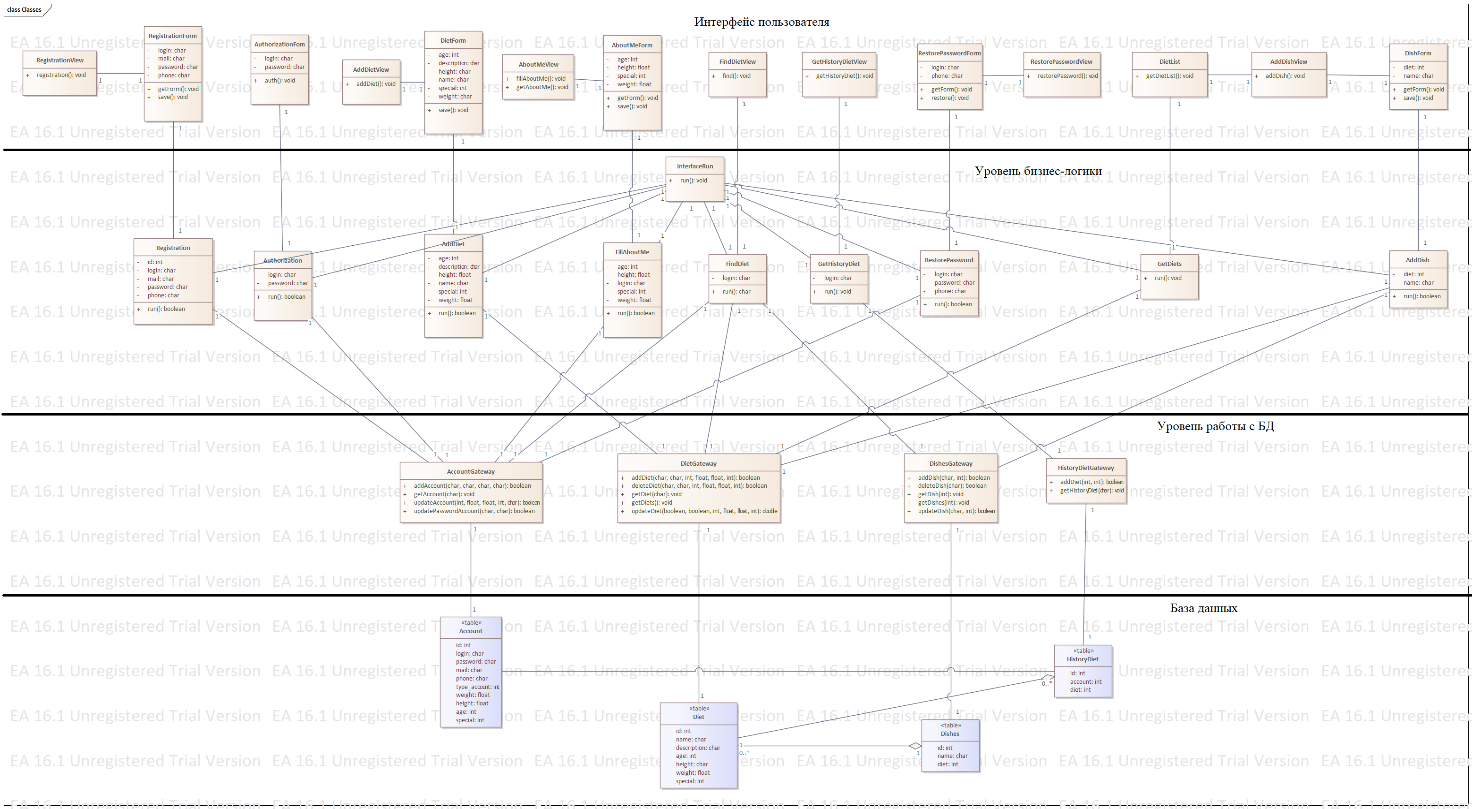


Диаграмма последовательностей для прецедента восстановление пароля:



Общая диаграмма классов:



**Программный код**

**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):  super().\_\_init\_\_()  self.login = login  self.password = password  self.mail = mail  self.phone = phone  def run(self):  return AccountGateway().addAccount(self.login, self.password, self.phone, self.mail) |

Класс Authorization для выполнения сценария авторизации:

|  |
| --- |
| class Authorization(InterfaceRun):  def \_\_init\_\_(self, login, password):  super().\_\_init\_\_()  self.login = login  self.password = password  def run(self):  acc = AccountGateway().getAccount(self.login)  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):  super().\_\_init\_\_()  self.name = name  self.description = description  self.age = age  self.height = height  self.weight = weight  self.special = special  def run(self):  return DietGateway().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):  super().\_\_init\_\_()  self.age = age  self.height = height  self.weight = weight  self.special = special  self.login = login  def run(self):  return AccountGateway().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):  super().\_\_init\_\_()  self.login = login  def run(self):  account = AccountGateway().getAccount(self.login)  diets = DietGateway().getDiets()  # Подбор диеты  diet\_id = diets[0]["id"]  diet\_name = diets[0]["name"]  # Вернуть блюда + название диеты  return diet\_name, DishesGateway().getDishes(diet\_id) |

Класс GetHistoryDiet для выполнения сценария получения истории диет:

|  |
| --- |
| class GetHistoryDiet(InterfaceRun):  def \_\_init\_\_(self, login):  super().\_\_init\_\_()  self.login = login  def run(self):  return HistoryDietGateway().getHistoryDiet(self.login) |

Класс RestorePassword – выполнения сценария восстановления пароля:

|  |
| --- |
| class RestorePassword(InterfaceRun):  def \_\_init\_\_(self, login, phone, password):  super().\_\_init\_\_()  self.login = login  self.phone = phone  self.password = password  def run(self):  AccountGateway().getAccount(self.login)  return AccountGateway().updatePasswordAccount(self.login, self.password) |

**Реализация классов Table Data Gateway**

Класс AccountGateway – шлюз таблицы данных для таблицы Account:

|  |
| --- |
| class AccountGateway:  def addAccount(self, login, password, phone, mail):  try:  cursor.execute("INSERT INTO Account (login, password, phone, mail) VALUES (?, ?, ?, ?)", (login, password, phone, mail))  except:  return False  return True  def updateAccount(self, age, height, weight, special, login):  try:  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:  cursor.execute("UPDATE Account SET password = ? WHERE login = ?", (password, login))  except:  return False  return True  def getAccount(self, login):  cursor.execute(  "SELECT login, password, phone, mail, type\_account, age, height, weight, special FROM Account WHERE login = ?",  (login,))  query = 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]})  return res |

Класс DietGateway– шлюз таблицы данных для таблицы Diet:

|  |
| --- |
| class DietGateway:  def addDiet(self, name, description, age, height, weight, special):  try:  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, name, description, age, height, weight, special, id\_diet):  try:  cursor.execute(  "UPDATE Diet SET name = ?, description = ?, age = ?, height = ?, weight = ?, special = ? WHERE id = ?",  (name, description, age, height, weight, special, id\_diet))  except:  return False  return True  def getDiet(self, name):  cursor.execute("SELECT name, description, age, height, weight, special FROM Diet WHERE name = ?", (name,))  query = 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 getDiets(self):  cursor.execute("SELECT id, name, description, age, height, weight, special FROM Diet")  query = 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:  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 addDish(self, name, diet):  try:  cursor.execute("INSERT INTO Dishes (name, diet) VALUES (?, ?)", (name, diet))  except:  return False  return True  def updateDish(self, name, id\_dish):  try:  cursor.execute("UPDATE Dishes SET name = ? WHERE id = ?", (name, id\_dish))  except:  return False  return True  def getDish(self, id\_dish):  cursor.execute("SELECT name FROM Dishes WHERE id = ?", (id\_dish,))  query = cursor.fetchall()  res = []  for str\_db in query:  res.append({"name": str\_db[0]})  return res  def getDishes(self, diet):  cursor.execute("SELECT name FROM Dishes WHERE diet = ?", (diet,))  query = cursor.fetchall()  res = []  for str\_db in query:  res.append({"name": str\_db[0]})  return res  def deleteDish(self, name):  try:  cursor.execute("DELETE FROM Dish WHERE name = ?", (name,))  except:  return False  return True |

Класс HistoryDietGateway– шлюз таблицы данных для таблицы HistoryDiet:

|  |
| --- |
| class HistoryDietGateway:  def addDiet(self, account, diet):  try:  cursor.execute("INSERT INTO HistoryDiet (account, diet) VALUES (?, ?)", (account, diet))  except:  return False  return True  def getHistoryDiet(self, login):  cursor.execute( "SELECT Diet.name FROM HistoryDiet JOIN Diet on HistoryDiet.diet = Diet.id JOIN Account on Account.id = HistoryDiet.account WHERE Account.login = ?", (login,))  query = cursor.fetchall()  res = []  for str\_db in query:  res.append({"name": str\_db[0]})  return res |

**Список источников**

1. Конспект лекций по курсу Технологии разработки программного обеспечения.
2. Мартина Фаулера «Архитектура корпоративных программных приложений»