**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ITMO University**

**Домашняя работа**

**По дисциплине** Проектирование инфокоммуникационных систем

**Тема работы** Реализация индивидуального задания

**Обучающийся** Морозова Яна Александровна

**Факультет** факультет инфокоммуникационных технологий

**Группа** К3320

**Направление подготовки** 11.03.02 Инфокоммуникационные технологии и системы связи

**Образовательная программа** Программирование в инфокоммуникационных системах

**Обучающийся** 20.12.2024  Морозова Я.А.

(дата) (подпись) (Ф.И.О.)

**Руководитель**  Осипов Н.А.

(дата) (подпись) (Ф.И.О.)

**СОДЕРЖАНИЕ**

[1. ЦЕЛЬ РАБОТЫ 3](#_Toc183261285)

[2. ХОД РАБОТЫ 3](#_Toc183261286)

[2.1 Реализация начальной фазы проекта 3](#_Toc183261287)

[2.2 Создание прецедентов на уровне элементарных бизнес-процессов (EBP) 7](#_Toc183261288)

[2.3 Построение и исследование моделей сценария использования (Use Case) 9](#_Toc183261289)

[ВЫВОД 13](#_Toc183261290)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc183261291)

# ЦЕЛЬ РАБОТЫ

Разработать уточненную (после проведения практических занятий) диаграмму классов. Спроектировать классы как можно подробнее. Применить шаблоны GoF в тех случаях, где это полезно. Программно реализовать спроектированную модель.

# ХОД РАБОТЫ

## Разработка диаграммы классов

**Атрибуты и методы классов**

1 Класс User

* Описание: представляет пользователя системы.
* Атрибуты: user\_id (int): Уникальный идентификатор пользователя. Name (str): Имя пользователя. Email (str): Электронная почта пользователя.
* Методы: \_\_init\_\_(self, user\_id, name, email): Конструктор, инициализирующий объект. \_\_str\_\_(self): Возвращает строковое представление пользователя.

2 Класс UserFactory

* Описание: Фабрика для создания объектов класса User.
* Методы: create\_user(user\_type, user\_id, name, email): Создает пользователя заданного типа.

3 Класс RecipeDatabase

* Описание: хранит и управляет базой данных рецептов (Singleton).
* Атрибуты: recipes (list): Список объектов Recipe.
* Методы: \_\_new\_\_(cls, \*args, \*\*kwargs): Реализует Singleton, возвращает единственный экземпляр. add\_recipe(self, recipe): Добавляет рецепт в базу. get\_all\_recipes(self): Возвращает список всех рецептов.

4 Класс Recipe

* Описание: представляет данные одного рецепта.
* Атрибуты: title (str): Название рецепта. ingredients (list): Список ингредиентов. instructions (str): Описание процесса приготовления.
* Методы: \_\_init\_\_(self, title, ingredients, instructions): Конструктор, инициализирует рецепт. get\_details(self): Возвращает строковое представление рецепта.

5 Класс RecipeDecorator

* Описание: Декоратор для добавления функциональности к рецепту.
* Атрибуты: recipe (Recipe): Оборачиваемый рецепт.
* Методы: \_\_init\_\_(self, recipe): Конструктор, принимает объект Recipe. get\_details(self): Возвращает данные об обернутом рецепте.

6 Класс FavoriteRecipeDecorator (наследник RecipeDecorator)

* Описание: Декоратор для добавления статуса "Избранное" рецепту.
* Методы: get\_details(self): возвращает данные об обернутом рецепте с пометкой "Избранное".

7 Класс SearchStrategy (абстрактный)

* Описание: определяет интерфейс для стратегий поиска.
* Методы: search(self, query, recipes): Абстрактный метод поиска.

8 Класс SearchByTitle (наследник SearchStrategy)

* Описание: Реализация стратегии поиска по названию рецепта.
* Методы: search(self, query, recipes): Ищет рецепты по названию.

9 Класс SearchByIngredient (наследник SearchStrategy)

* Описание: Реализация стратегии поиска по ингредиентам.
* Методы: search(self, query, recipes): Ищет рецепты по ингредиентам.

10 Класс RecipeSearchController

* Описание: Контроллер для управления стратегиями поиска.
* Атрибуты: strategy (SearchStrategy): Текущая стратегия поиска.
* Методы: \_\_init\_\_(self, strategy): Конструктор, принимает начальную стратегию. set\_strategy(self, strategy): Устанавливает новую стратегию поиска. search(self, query, recipes): Выполняет поиск по текущей стратегии.

11 Класс RecipeNotifier

* Описание: управляет подпиской пользователей на уведомления о новых рецептах.
* Атрибуты: subscribers (list): Список подписчиков (User).
* Методы: subscribe(self, user): Добавляет подписчика. notify(self, recipe): Уведомляет подписчиков о новом рецепте.

12 Класс UserInterface

* Описание: предоставляет интерфейс взаимодействия с пользователем.
* Атрибуты: user (User): Пользователь, связанный с интерфейсом. recipe\_db (RecipeDatabase): Экземпляр базы данных рецептов. search\_controller (RecipeSearchController): Контроллер для поиска.
* Методы: \_\_init\_\_(self, user, recipe\_db, search\_controller): Конструктор. display\_recipes(self): Показывает все рецепты из базы. search\_recipes(self, query): Выполняет поиск рецептов по запросу пользователя.

**Шаблоны:**

* Singleton для класса RecipeDatabase — гарантирует единый объект базы данных.
* Strategy для RecipeSearchController — реализует разные алгоритмы поиска.
* Observer для оповещения пользователей о новых рецептах.
* Decorator для расширения функциональности рецептов.
* Factory для создания пользователей.

# ВЫВОД

В ходе выполнения лабораторной работы были определены основные классы, их атрибуты и методы для создания системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Diagrams: официальный сайт. – Санкт-Петербург. – URL: <https://www.diagrams.net> (дата обращения: 01.11.2022)