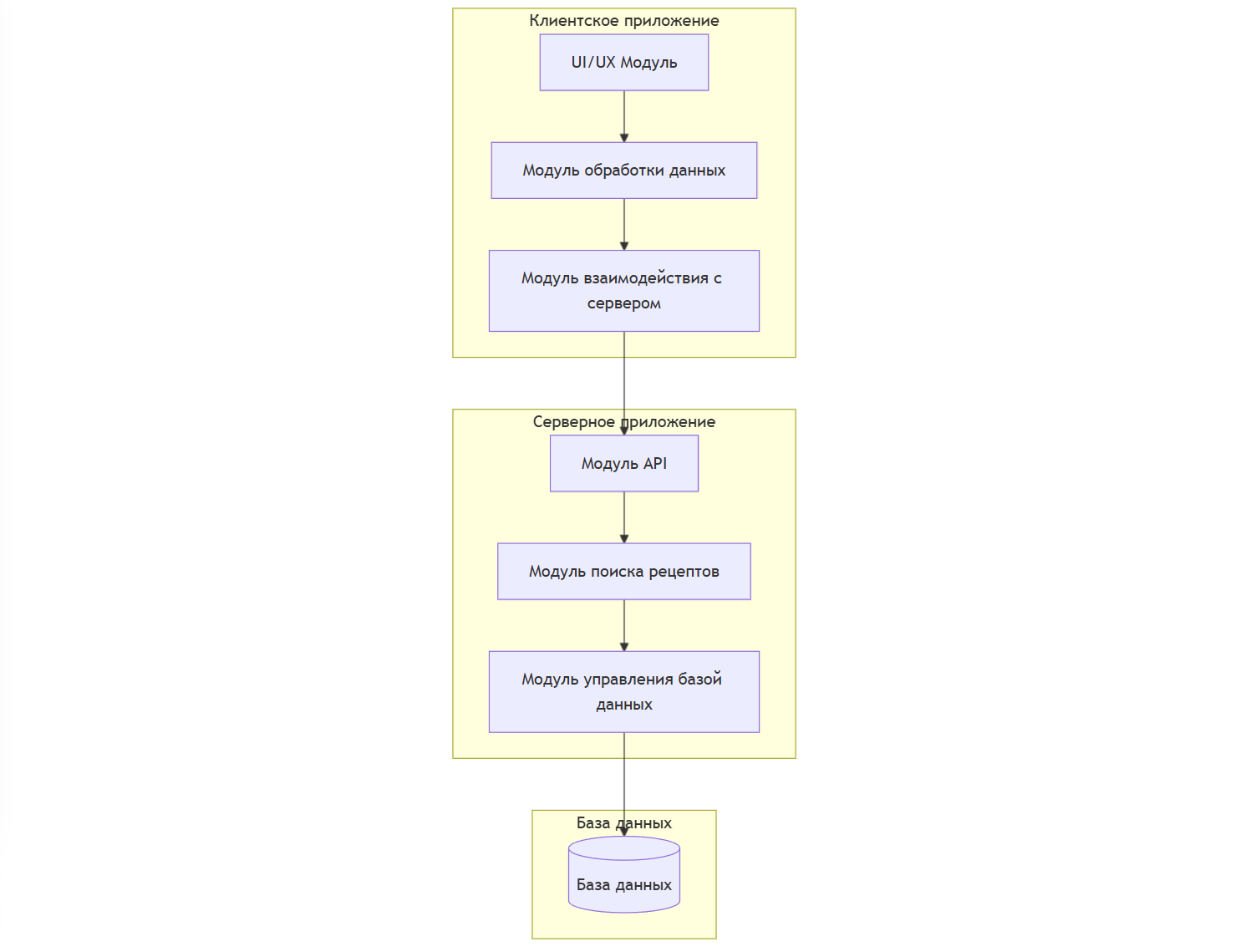
### Часть 1: Проектирование общей архитектуры

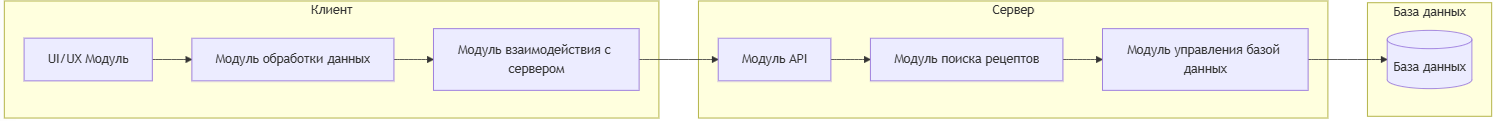
**Общая архитектура приложения "Готовь из того, что есть"**

Приложение представляет собой клиент-серверную архитектуру с использованием мобильного интерфейса на стороне клиента и серверной базы данных для хранения рецептов.

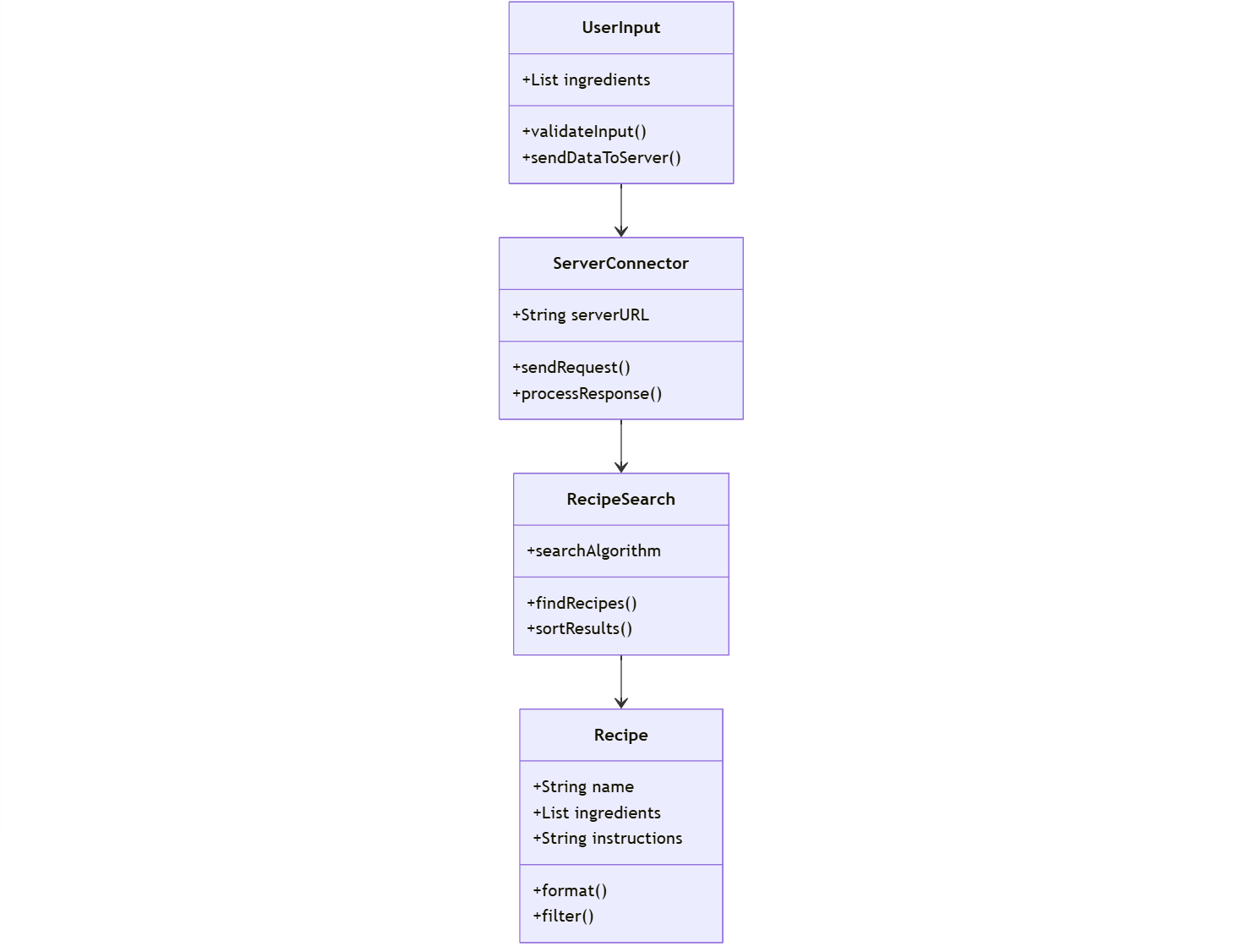
### Диаграмма архитектуры приложения:



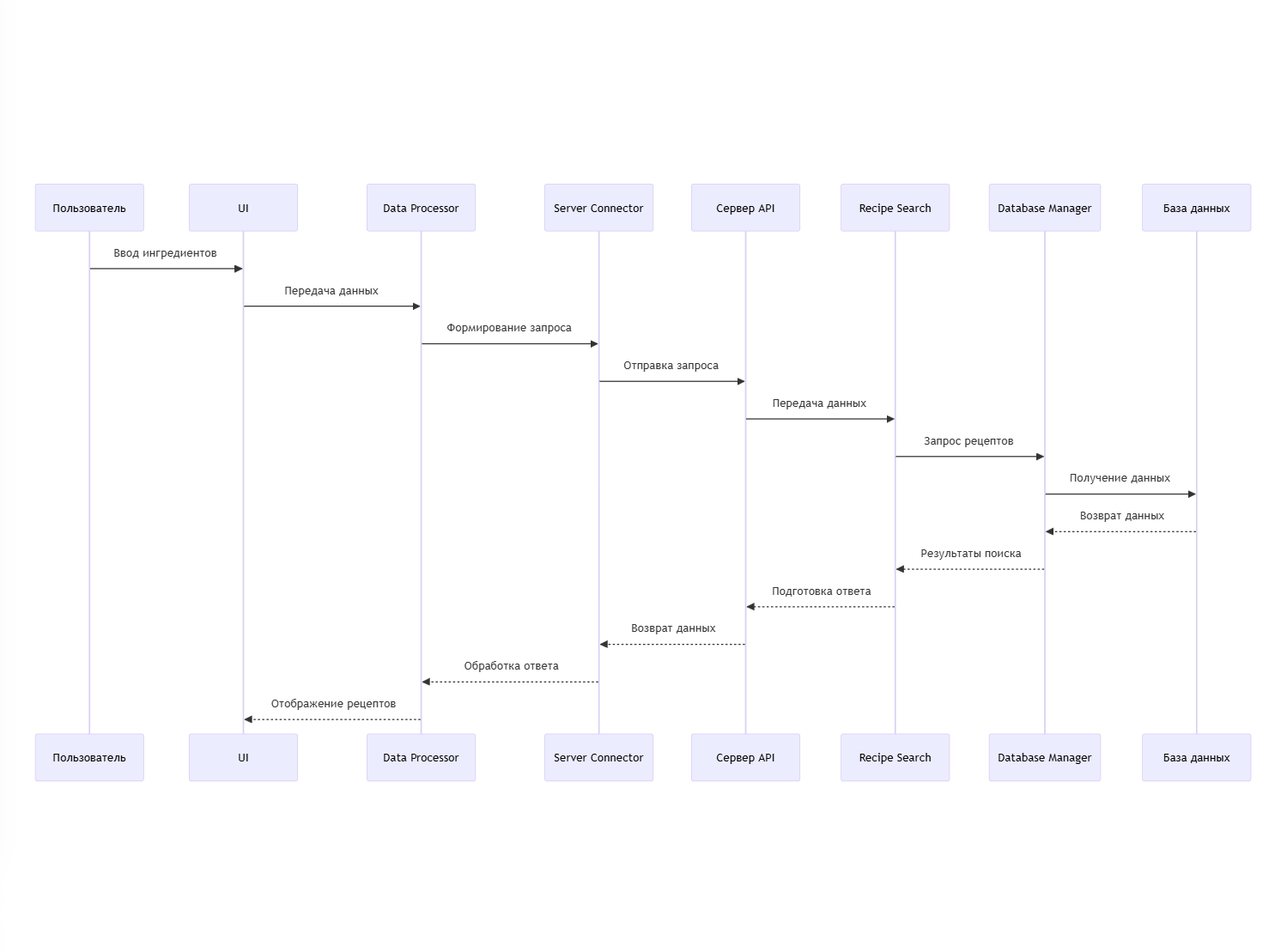
### 1. Диаграмма компонентов (Component Diagram)



### 2. Диаграмма классов (Class Diagram)



### 3. Диаграмма последовательностей (Sequence Diagram)



### Часть 2: Определение структуры модулей

### 1. Определение модулей:

### - UI/UX Module: Обеспечивает взаимодействие с пользователем. - Data Processing Module: Обрабатывает данные, введенные пользователем. - Server Connector Module: Передает запросы серверу и обрабатывает ответы. - API Module: Управляет входящими запросами и их маршрутизацией. - Recipe Search Module: Выполняет подбор рецептов по заданным ингредиентам. - Database Manager Module: Обеспечивает доступ к базе данных.

### 2. Разработка API интерфейсов:

### Примеры API методов: - POST /ingredients: Отправка списка ингредиентов. - GET /recipes: Получение списка рецептов на основе ингредиентов. - GET /recipe/{id}: Получение деталей рецепта.

### Часть 3: Проектирование компонентов

### Диаграмма компонентов для Recipe Search Module:

### 

### Пример: Recipe Search Module

### - Компоненты: - RecipeController: Принимает запросы от API. - RecipeService: Обрабатывает бизнес-логику. - RecipeRepository: Взаимодействует с базой данных.

### RecipeController --> RecipeService --> RecipeRepository RecipeRepository --> DB[(База данных)]

### Часть 4: Документирование архитектуры

### 1. Архитектурный документ:

### Общее описание: Приложение делится на три основных слоя: клиент, сервер и база данных. Используется клиент-серверная архитектура для удобства масштабирования и поддержки. Схемы и диаграммы: Включают общую архитектуру, компоненты, классы и последовательности взаимодействий. Функциональность модулей: Модули разделены по задачам: от обработки пользовательского ввода до взаимодействия с базой данных.

### 2. Обоснование решений:

### Архитектурный стиль: Клиент-серверный подход подходит для обработки больших объемов данных и централизованного управления. Структура модулей: Гибкость и масштабируемость достигаются за счет четкого разделения задач между модулями.