2.1 Архитектурная организация приложения: Для обеспечения модульности, тестируемости и поддерживаемости приложения применена чистая архитектура. Данный подход позволяет четко разделить ответственности между слоями приложения (presentation, domain, data), что упрощает внедрение изменений и добавление нового функционала. [Ссылки на статьи/книги про чистую архитектуру].

2.2 Взаимодействие с серверной частью и обработка данных: Обмен данными между клиентским приложением и сервером осуществляется посредством REST API. Для преобразования данных в формат, подходящий для передачи по сети, используются механизмы сериализации и десериализации (например, с использованием JSON). Особое внимание уделено обработке потенциальных ошибок и асинхронных операций для обеспечения стабильной работы приложения. [Ссылки на документацию по REST API и используемым библиотекам].

2.3 Связь бизнес-логики с пользовательским интерфейсом: Отделение бизнес-логики от UI, Использование паттернов управления состоянием,Обработка событий и обратной связи с пользователем,Реактивность интерфейса и управление состоянием, Реализация связи между уровнем представления (UI) и бизнес-логикой приложения построена с использованием . Этот подход обеспечивает эффективное обновление UI в ответ на изменения данных и состояния приложения, сохраняя при этом четкое разделение ответственностей и упрощая тестирование. [Ссылки на документацию по выбранному паттерну].