ПРИМЕНЕНИЕ ЧИСТОЙ АРХИТЕКТУРЫ В ИНФОРМАЦИОННОЙ СИСТЕМЕ ДЛЯ АВТОМАТИЗАЦИИ УЧЕБНОГО ПРОЦЕССА В ОБРАЗОВАТЕЛЬНОЙ ОРГАНИЗАЦИИ

Бурцев Т.Р., студент,
Тазетдинов Б.И., к.ф.-м.н., доцент,
Бирский филиал УУНиТ, г. Бирск, Россия

Аннотация: В данной научной статье рассматривается применение чистой архитектуры в информационной системе, разработанной для автоматизации учебного процесса в образовательной организации. Опираясь на принципы чистой архитектуры, мы предлагаем эффективное решение, которое позволяет разделить бизнес-логику системы от ее технической реализации. Это обеспечивает гибкость и масштабируемость системы, а также упрощает его поддержку и развитие.

Ключевые слова: чистая архитектура, информационная система, автоматизация учебного процесса, образовательная организация

Введение информационных технологий в образовательную среду было одним из основных направлений развития современной образовательной системы. Автоматизация учебного процесса позволяет улучшить качество обучения, упростить его организацию и оценку, а также дает возможность педагогам и учащимся использовать новые методы обучения и интерактивные материалы. Однако, при разработке информационной системы для автоматизации учебного процесса, важно продумывать его архитектуру таким образом, чтобы она была гибкой, масштабируемой и поддерживаемой

Чистая архитектура - это концепция разработки, которая ставит целью создание простого, гибкого и расширяемого приложения. Она предлагает определенные принципы и правила построения архитектуры приложения, которые помогают избежать излишней сложности и связанной с ней путаницы.

Один из главных принципов чистой архитектуры - это строгое разделение приложения на слои. В основе этого подхода лежит идея о том, что каждый слой должен быть отвечать только за конкретную задачу и не должен зависеть от деталей реализации других слоев.

Первый слой в чистой архитектуре - это слой представления. Он отвечает за отображение данных и взаимодействие с пользователем. В веб-приложении это может быть фронтенд, который отвечает за визуальное представление данных и обработку пользовательских действий.

Второй слой - это слой бизнес-логики. Он содержит основную логику приложения и обрабатывает данные, полученные от слоя представления. Бизнес-логика веб-приложения может включать в себя проверку и обработку данных, осуществление бизнес-процессов и другие задачи, не связанные с конкретным интерфейсом.

Третий слой - это слой доступа к данным. Он отвечает за работу с базой данных или другими источниками данных. Веб-приложение может использовать ORM (объектно-реляционное отображение), чтобы обеспечить простоту и прозрачность доступа к данным.

Важным аспектом чистой архитектуры является применение принципа инверсии зависимостей (Dependency Inversion Principle, DIP). Он предлагает, чтобы зависимости между слоями были инвертированы, позволяя более гибко менять реализацию слоев и делать их независимыми друг от друга.

Преимущества чистой архитектуры веб-приложения включают легкость разработки и сопровождения, улучшение тестируемости и возможность масштабирования приложения. Благодаря четкому разделению слоев, изменение одной части приложения не требует модификации остальных, что снижает риск возникновения ошибок и упрощает добавление новых функций.

Однако, стоит отметить, что применение чистой архитектуры может потребовать больше времени и усилий при начале разработки. Определение и построение слоев и зависимостей требует дополнительного анализа и

планирования. Однако, эти усилия окупятся в будущем, когда приложение станет более гибким и легко изменяемым.

В данной статье рассмотрим несколько основных компонентов информационной системы для автоматизации учебного процесса в образовательной организации. Это система оценки, формировании учебного плана, классов, записи учеников и работников школы в одну систему и авторизации в ней.

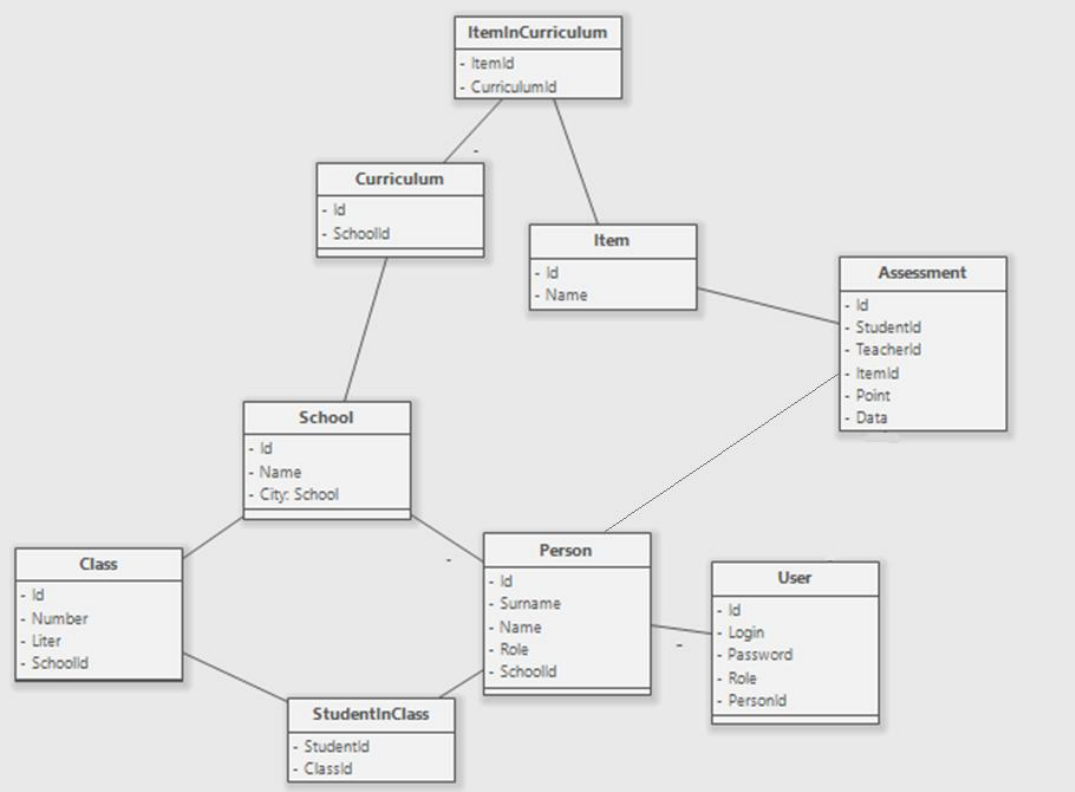

Рис. 1 – Даталогическая модель базы данных

Рассмотрим пример реализации чистой архитектуры при разработки информационной системы с использованием технологии ASP.NET Core для серверной части, Blazor WebAssembly для клиентской части, JWT-Token для авторизации и SQLite как база данных.

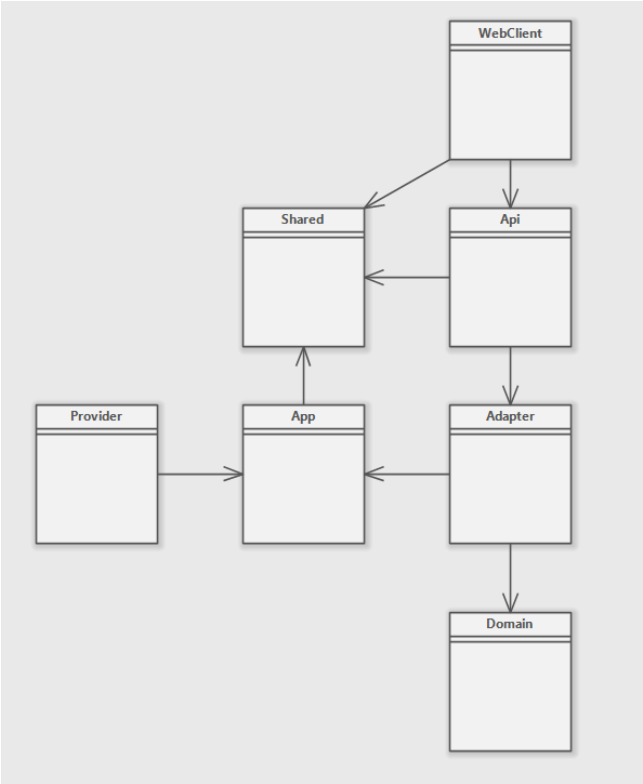

Рис. 2 – Уровни архитектуры приложения

Domain – представлен в виде библиотеки классов, в которой хранятся классы, чьи экземпляры записываются в базу данных. Доступ к Domain имеет только Adapter.

Adapter – библиотека классов, отвечающая за абстрагирование уровня данных от остальной части приложения и предоставления способа выполнения операций с данными, не зная, как хранятся или извлекаются данные. *GenericRepository* – паттерн репозитория, методы которого непосредственно осуществляют добавление и извлечение данных из БД. AuthRepository отвечает за добавление и извлечение данных для авторизации в системе из БД. *UnitWork* сохраняет изменения в базе данных.

Shared– библиотека классов, в чьих экземплярах передаются данные между клиентом (ClientUI) и сервером (Api). Классы *Dto* являются копией классов Entity из Domain, за исключением полей, информация из которых не должна отображаться пользователю или напротив, не должна хранится в базе данных, а участвует лишь в бизнес-логике. Класс *Response* хранит в себе экземпляр класса Dto, а также информацию об ошибке, если таковая возникла.

Page используется для постраничного вывода данных из БД, в нем хранится массив данных и информация о номере страницы и количестве элементов в ней. Response и Page используются лишь для вывода данных клиенту (пользователю).

App библиотека классов, которая содержит всю бизнес-логику. Задача библиотеки – абстрагировать бизнес-логику от клиента, одновременно упрощая приложение и делая будущие изменения более легкими. Именно в классах *Interactor* хранится вся бизнес-логика приложения. Классы *Mappers* вспомогательные классы для Interactor. Они преобразуют классы Entity в аналогичные классы из Dto, для отображения данных клиенту и наоборот, из Dto в аналогичные Entity для отправки в repository с последующем добавлением данных в БД. В *Storage* хранятся интерфейсы AuthInterface и GenericInterface, которые реализуются в Adapter. Это сделано, чтобы избежать зацикленной зависимости уровней архитектуры и не дать доступ App к реализации логики хранения и извлечения данных.

Api сам сервер. Сервер использует Веб-платформу ASP.NET Core. В Api записаны *Controller* – группы методов получающие данные от клиента через json запросы. Т.к. вся бизнес-логика хранится на уровне App, а способы записи и чтения БД в Adapter, то Controller лишь вызывает и передает данные в Interactor, что дает возможность использовать любую Веб-платформу. Также в Api хранится информация о *миграции базы данных*.

Provider библиотека классов, хранящая группы методов, отвечающие за авторизацию путем JWT токенов.

WebClient интерфейс клиента. Для реализации интерфейса было использовано веб-приложение Blazor WebAssembly. Клиентская часть приложение формирует http запросы и отправляет их на сервер. По аналогии с веб-платформой, веб-приложение может быть любым, без изменений бизнес-логики.

В результате использования данной архитектуры приложение становится более гибким в разработке, модельным и легко поддерживаемым в будущем.

В заключение, чистая архитектура предоставляет способ построения гибкого, расширяемого и легко сопровождаемого приложения. Ее основные принципы - разделение на слои, инверсия зависимостей и простота тестирования - помогают избежать излишней сложности и сделать приложение более устойчивым к изменениям.

Литература

1. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер, 2021. 352 с.
2. Clean Architecture. URL: https://habr.com/ru/companies/otus/articles/732178/ (дата обращения: 15.11.23)
3. Чистая архитектура в ASP.NET Core Web API. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.e5eaf97a-6559e67c-83c9393b-74722d776562/https/www.c-sharpcorner.com/article/clean-architecture-in-asp-net-core-web-api/ (дата обращения: 16.11.23)
4. Приложение. Порядок организации и осуществления образовательной деятельности по основным общеобразовательным программам - образовательным программам начального общего, основного общего и среднего общего образования. URL: https://base.garant.ru/400663548/53f89421bbdaf741eb2d1ecc4ddb4c33/?ysclid=lp5gsyxnwe446481792 (дата обращения: 16.11.23)