**УДК 330.341.2**

# FEATURE-SLICED DESIGN КАК УНИВЕРСАЛЬНАЯ АРХИТЕКТУРА ДЛЯ FRONTEND-ПРОЕКТОВ

**Скоморохов М.И.[1]**

*[1]Уральский государственный Экономический Университет «УрГЭУ», Екатеринбург, e-mail: b.boy.matv@gmail.com*

**Эта статья посвящена архитектурной методологии для Frontend-проектов, известной как Feature-Sliced Design (FSD). Первая часть статьи прослеживает историю развития архитектур программного обеспечения, подчеркивая важность правильной структуры и организации кода. Автор подробно описывает особенности FSD, обсуждая его основные концепции и терминах, таких как "слои", "слайсы" и "сегменты". С особенным вниманием рассматриваются различные виды слоев, их назначение и взаимодействие. Затем автор демонстрирует применение FSD на реальном примере - разработке приложения для управления задачами. Описаны ключевые компоненты приложения, их функции и взаимодействия, а также приведена структура проекта. В заключительной части автор делает вывод о эффективности использования FSD в контексте персонального проекта и его положительном влиянии на стандартизацию процесса разработки, упрощение навигации по кодовой базе и облегчение адаптации новых сотрудников.**

Ключевые слова: Программное обеспечение, Архитектура ПО, Frontend разработка, Feature-Sliced Design, Структура проекта, Стандартизация в разработке ПО

# FEATURE-SLICED DESIGN AS UNIVERSAL ARCHITECTURE FOR FRONTEND PROJECTS

**Skomorokhov M.I.[1]**

*[1]Ural State University of Economics, Yakaterinburg, e-mail: b.boy.matv@gmail.com*

**This article is dedicated to an architectural methodology for Frontend projects, known as Feature-Sliced Design (FSD). The first part of the article traces the history of software architecture development, highlighting the importance of correct code structure and organization. The author detailedly describes the features of the FSD, discussing its core concepts and terms such as "layers", "slices", and "segments". Various types of layers, their purpose and interaction are examined with special attention. Then, the author demonstrates the application of FSD on a real-life example - the development of a task management application. The key components of the application, their functions and interactions, as well as the project's structure are described. In the concluding part, the author comments on the effectiveness of using FSD in the context of a personal project and its positive impact on the standardization of the development process, simplification of codebase navigation, and facilitating the adaptation of new team members.**

Keywords: Software, Software Architecture, Frontend Development, Feature-Sliced Design, Project Structure, Standardization in Software Development

В постоянно меняющемся ландшафте разработки интерфейсов поиск универсальной архитектуры, которая органично вписывалась бы в динамичные требования разнообразных проектов, остается первостепенной задачей [7]. На фоне этого стремления парадигма функционального дизайна становится многообещающим и универсальным решением.

С развитием и усложнением разработки программного обеспечения (далее – ПО) появилась нужда в стандартизированных решениях в области проектирования систем [8]. Еще в конце 60-х годов 20 века в научно-исследовательской работе Эдсгера Дейкстры [2] и Дэвида Парнаса [1] были заложены концепции архитектуры ПО. Эти ученые подчеркнули, что структура системы

ПО имеет важное значение и что построение правильной структуры — критически важно. Популярность изучения этой области возросла с начала 1990-х годов вместе с научно-исследовательской работой по исследованию архитектурных стилей (шаблонов), языков описания архитектуры, документирования архитектуры и формальных методов [4].

Долгое время архитектуры разрабатывались преимущественно для десктопного ПО, но с развитием интернета появилось новое направление разработки – Frontend [9]. Чаще всего Frontend разработчики разрабатывают пользовательские интерфейсы для веб-сайтов и веб-приложений, но они могут разрабатывать интерфейсы для мобильных и десктопных приложений [5]. Поскольку Frontend разработка достаточно молода, то в ней еще не успели образоваться архитектурные методологии и решения, которые закрепились бы в корпоративной разработке и стали стандартом индустрии.

Об одной из таких попыток пойдет речь в этой статье. Feature-Sliced Design (далее – FSD) – это архитектурная методология для Frontend-проектов [6]. FSD разработало сообщество российских разработчиков из разных компаний. Эту архитектурную методологию уже применяют в X5 Digital, Dodo Engineering, Samokat.tech, почтатех.

Разработка интерфейса корпоративного масштаба требует архитектурных парадигм, которые могут эффективно решать многогранные задачи, присущие сложным и развивающимся программным экосистемам. Функционально-ориентированный дизайн (FSD) возникает как последовательное решение, обеспечивающее структурированный подход к системной архитектуре, который соответствует динамичным требованиям современных предприятий.

Постоянно растущий масштаб и сложность корпоративных приложений требуют целостной архитектурной стратегии для обеспечения удобства обслуживания, масштабируемости и адаптивности. FSD, благодаря своему модульному и функционально-ориентированному дизайну, удовлетворяет этим требованиям, облегчая детальную организацию компонентов кодовой базы. Такая модульность не только повышает удобство сопровождения кода, но и создает среду, благоприятствующую сотрудничеству между командами разработчиков, работающими над различными функциями.

Более того, предприятия часто сталкиваются с необходимостью адаптации своих программных систем к меняющимся бизнес-требованиям и технологическому ландшафту. Функционально-ориентированный подход FSD не только упрощает процесс интеграции новых функций, но и облегчает удаление или модификацию существующих функций, не вызывая серьезных сбоев в работе системы в целом. Такая адаптивность является критически важным активом в корпоративных сценариях, где гибкость и быстрота реагирования на изменения имеют первостепенное значение.

В контексте крупномасштабных команд разработчиков, характерных для корпоративных сред, FSD способствует параллельной разработке, разделяя функции на управляемые блоки. Это приводит к сокращению узких мест в разработке и ускорению вывода на рынок новых функциональных возможностей. Присущая FSD модульность также способствует улучшению процессов тестирования, позволяя проводить целенаправленное и эффективное тестирование отдельных функций, тем самым повышая общую надежность системы.

Целью текущей работы является применение Feature-Sliced Design в условиях персонального проекта. Задачи статьи таковы:

– Изучить историю появления архитектур ПО;
– Рассмотреть архитектурную методологию Feature-Sliced Design;
– Применить FSD на персональном проекте.

Разберем методологию подробно. Как было упомянуто ранее, FSD – это именно архитектурная методология для фронтенд проектов. В отличии от строгой архитектуры, архитектурная методология представляет собой свод правил и соглашений по организации кода. Главная цель методологии — сделать проект понятным и структурированным, особенно в условиях регулярного изменения требований бизнеса. FSD подходит в следующих случаях:

– Эта методология исключительно для фронтенда;
– Для маленьких проектов она может быть избыточна, но и для них её можно использовать;
– Для огромных проектов FSD может выступить в качестве отправной точки.

Одна из ключевых идей FSD – структура проекта. Проект на FSD состоит из слоев (layers), каждый слой состоит из слайсов (slices) и каждый слайс состоит из сегментов (segments), визуализация структуры представлена на рисунке 1.

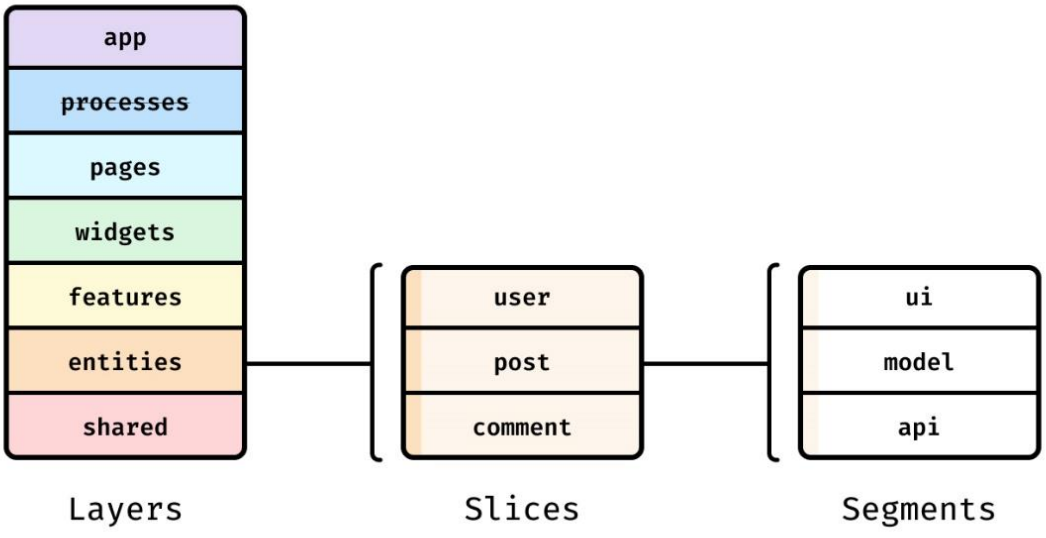


Рисунок 1 – структура проекта на FSD

Рассмотрим каждый термин подробнее. Слои задают основу любого проекта на FSD, они стандартизированы. Каждый слой может использовать только слои ниже себя. Сейчас разработчики методологии выделили 6 слоев.

1) В shared лежит переиспользуемый код, не имеющий отношения к специфике приложения или бизнеса. Это могут быть UIKit, библиотеки, API.

2) entities хранит в себе бизнес-сущности. Например, Task или User.

3) features содержит код, отвечающий за взаимодействия с пользователем, действия, которые несут бизнес-ценность для пользователя. К примеру, AddTask или FilterTasks.

4) Widgets – это композиционный слой для соединения сущностей и фич в самостоятельные блоки. Такими блоками могут быть TaskList, Header и Footer.

5) pages также является композиционным слоем, но для сборки полноценных страниц из сущностей, фич и виджетов.

6) app содержит настройки, стили, провайдеры для всего приложения, инициализирует его.

7) Архитектура регулярно обновляется и в начале лета слой processes удалили, он отвечал за сложные сценарии, покрывающие несколько страниц. Например, авторизация или оформление заказа.

Внутри слоев лежат слайсы, разделяющие код по предметной области. Слайсы группируют логически связанные модули, что облегчает навигацию по кодовой базе. Слайсы не могут использовать другие слайсы на том же слое, что обеспечивает высокий уровень связности при низком уровне зацепления.

Сами слайсы состоят из сегментов. Это маленькие модули, главная задача которых — разделить код внутри слайса по техническому назначению. Самые распространенные сегменты — ui, model (store, actions), api и lib (utils/hooks), но могут быть и другие.

Далее рассмотрим пример приложения на FSD. Приложение представляет собой список задач [3]. В качестве технологий были использованы:

- React;
- Redux Toolkit;
- TS;
- SCSS.

Слой app хранит в себе инициализацию проекта, логику перехода по страницам. В нём располагаются глобальные стили, которые применяются на всё приложение (например, шрифты и css-ресет), и компонент App, который хранит в себе логику маршрутизации по страницам приложения. Слой pages содержит компоненты, отвечающие за отрисовку страниц - главной страницы, страницы "О приложении", страницы с 404 ошибкой, а также Layout для

вышеперечисленных страниц. Каждый компонент pages состоит из виджетов, фичей и сущностей. О них далее.

Слой widgets содержит своего рода строительные блоки для страниц, в приложении были реализованы следующие виджеты: подвал, шапка приложения и TaskList. Последний отрисовывает массив задач. Слой features содержит функции приложения – добавить, изменить или удалить задачу, отобразить статистику по задачам, отфильтровать их. В этих компонентах преимущественно лежит логика по взаимодействию с менеджером состояния, который представляет собой глобальное хранилище. Слой entities содержит бизнес-сущности, в контексте примера – это сущность "Задача". Она содержит папку model, которая отвечает за работу с менеджером состояний, и папку ui, которая ответственна за отображение конкретной задачи. Слой shared хранит в себе переиспользуемые компоненты, в данном случае Dropdown и файлы стилей, содержащие css-переменные, шаблоны и mixins. Они не уникальны для приложения и могут быть использованы в других приложениях, поэтому эти компоненты лежат в shared. Пример проекта представлен на рисунке 2.

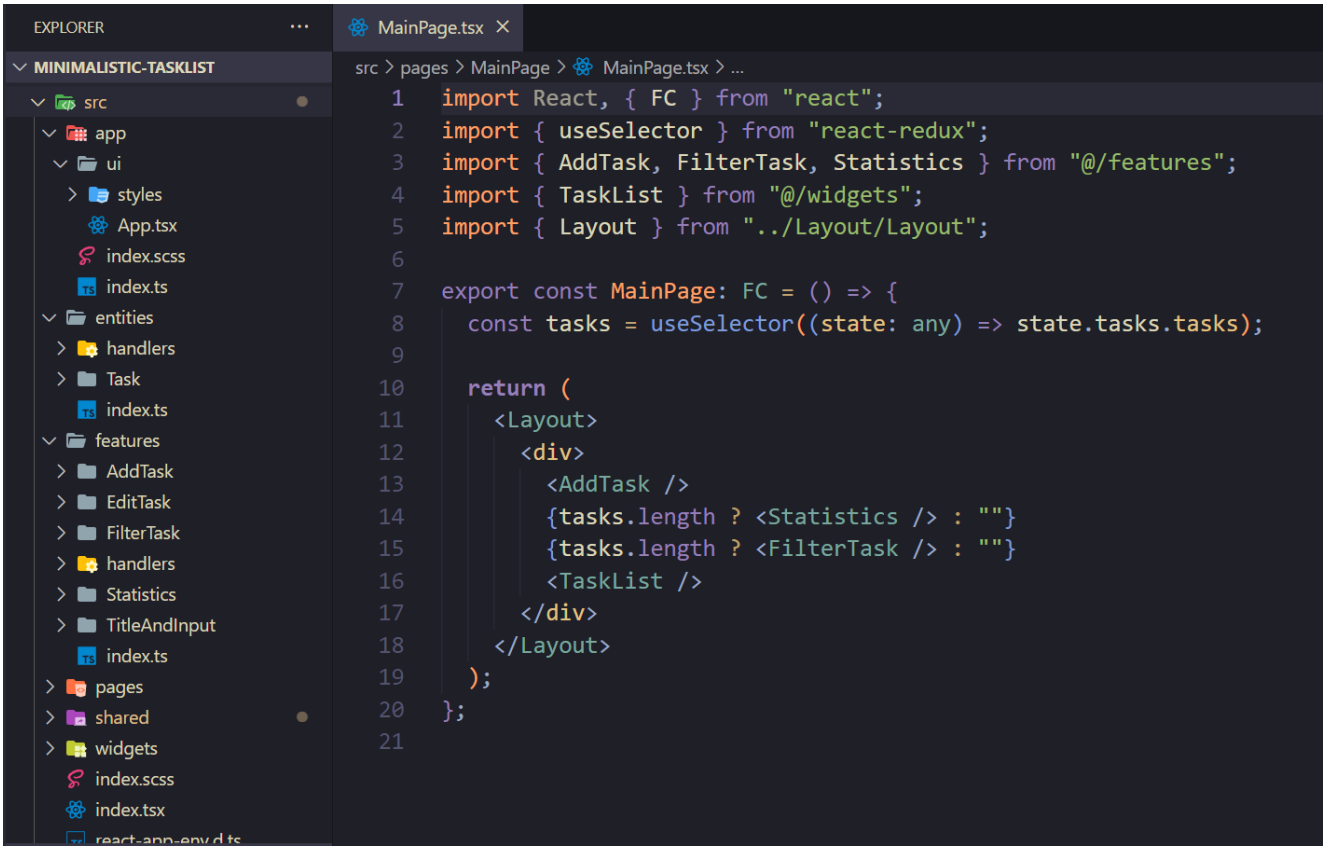


Рисунок 2 – пример структуры проекта на FSD

Методология Feature-Sliced Design (FSD), описанная в представленном тексте, представляет структурированный и модульный подход к архитектуре интерфейса, демонстрируя его применимость с помощью комплексной системы уровней. Расслоение на отдельные уровни —

общие, сущности, функции, виджеты, страницы и приложения — воплощает систематическую организацию, которая выходит за рамки традиционных монолитных архитектур.

Общий уровень инкапсулирует повторно используемые компоненты кода, лишенные специфики приложения или бизнеса, иллюстрируя парадигматический сдвиг в сторону инкапсуляции общих функциональных возможностей, таких как UIKit, библиотеки и API. Этот базовый уровень обеспечивает стандартизированную и модульную основу для всего проекта FSD, способствуя повторному использованию и облегчая техническое обслуживание.

Уровень сущностей, содержащий бизнес-объекты, такие как задача или пользователь, воплощает основные структуры данных, лежащие в основе бизнес-логики приложения. Такое разделение задач позволяет провести четкое разграничение между представительством бизнес-структур и функциональными возможностями, связанными с взаимодействием с пользователями, что способствует ясности и удобству обслуживания.

Функциональный уровень становится основой для кода взаимодействия с пользователем, инкапсулирующего действия, которые обеспечивают ценность бизнеса для конечных пользователей. Этот уровень олицетворяет согласование усилий по разработке с функциональными возможностями, ориентированными на пользователя, демонстрируя акцент на ясности и согласованности в организации кода.

Слой виджетов служит композиционным слоем, облегчающим сборку независимых блоков, представляющих сущности и функциональные возможности, в связные элементы, такие как список задач, верхний и нижний колонтитулы. Такая модульная композиция повышает расширяемость и ремонтопригодность кодовой базы.

Страницы, еще один композиционный уровень, организует создание полных страниц из сущностей, функций и виджетов. Такое иерархическое расположение обеспечивает систематическую организацию компонентов, способствуя масштабируемости и простоте разработки.

Уровень приложения охватывает конфигурации, стили и провайдеры в масштабах всего проекта, выступая в качестве общего инициализатора для всего приложения. Этот уровень инициализирует и настраивает приложение, предоставляя глобальные стили, которые пронизывают весь проект.

Динамичный характер FSD подчеркивается его эволюцией, примером которой является удаление слоя процессов. Эта адаптивная архитектура отражает способность реагировать на меняющийся ландшафт разработки интерфейсов, обеспечивая постоянную актуальность и эффективность методологии.

Внутри каждого уровня фрагменты дополнительно разделяют кодовую базу, группируя логически связанные модули для улучшения навигации. Запрет фрагментам в пределах одного слоя использовать друг друга обеспечивает высокий уровень когезии при низком уровне сцепления, что является фундаментальным принципом разработки программного обеспечения.

Срезы, в свою очередь, состоят из сегментов, небольших модулей, предназначенных для конкретных технических целей. Универсальность этих сегментов, охватывающих пользовательский интерфейс, модель (хранилище, действия), api и lib (утилиты/перехватчики), подчеркивает приверженность FSD решению разнообразных технических аспектов в рамках модульной структуры.

Практическое применение FSD проиллюстрировано на примере приложения со списком задач, использующего такие технологии, как React, Redux Toolkit, TypeScript (TS) и SCSS. Пошаговая разбивка проясняет сложное взаимодействие принципов FSD, демонстрируя, как каждый уровень вносит свой вклад в общую согласованность и функциональность приложения.

В заключение, функционально-ориентированный дизайн предстает как надежная и адаптивная архитектурная парадигма, предлагающая систематический и модульный подход к разработке интерфейса. Его многоуровневая структура в сочетании с принципами срезов и сегментов не только улучшает организацию кода, но и способствует возможности повторного использования, сопровождения и адаптации, что делает его привлекательным выбором для современных предприятий, ориентирующихся в сложностях крупномасштабных интерфейсных проектов.

Таким образом, в этой статье было рассмотрено применение архитектурного паттерна Feature-Sliced Design в условиях персонального проекта. В ходе статьи были выполнены поставленные цель и задачи. Была кратко рассмотрена история появления архитектур в сфере создания ПО, была рассмотрена архитектурная методология FSD и было создано приложение с применением этой методологии. Данная архитектура помогает облегчить процесс адаптации новых сотрудников, помогает системно думать об архитектуре Frontend проектов и стандартизирует процесс разработки ПО, давая рекомендации по разработке и упрощая навигацию по кодовой базе проекта.

1. Daniel M. Software Fundamentals: Collected Papers by David L. Parnas – Addison-Wesley Professional: 2001 – 688 p.

2. Архив Эдсгера Дийкстры. [Электронный ресурс] URL: https://www.cs.utexas.edu/users/EWD/ (дата обращения: 02.10.2023).

3. Персональный проект Скоморохова Матвей на GitHub. [Электронный ресурс] URL: https://github.com/MoneyTreesIsThePerfectPlaceForShade/good-task-m.a.a.d-list (дата обращения: 09.09.2023).

4. Сборник What Is Your Definition of Software Architecture. [Электронный ресурс] URL: https://insights.sei.cmu.edu/library/what-is-your-definition-of-software-architecture/ (дата обращения: 05.11.2023).
5. Статья компании OTUS на Хабр о Frontend разработке. [Электронный ресурс] URL: https://habr.com/ru/companies/otus/articles/674748/ (дата обращения: 23.09.2023).
6. Feature-Sliced Design. [Электронный ресурс] URL: https://feature-sliced.design/ru/ (дата обращения: 13.11.2023).
7. Подшивалова, М. В. Управление инновационным потенциалом малых предприятий высокотехнологичных отраслей / М. В. Подшивалова, С. К. Алмршед // Управленец. – 2021. – Т. 12, № 4. – С. 16-27. – DOI 10.29141/2218-5003-2021-12-4-2. – EDN ESMSPY.
8. Соловьева, И. А. Модель формирования эффективных команд для реализации инновационной деятельности предприятия / И. А. Соловьева, И. А. Мостовщикова // Journal of New Economy. – 2021. – Т. 22, № 2. – С. 110-133. – DOI 10.29141/2658-5081-2021-22-2-6. – EDN ZWMGQQ.
9. Кислицын, Е. В. Исследование архитектуры сценариев для создания аналитической отчетности / Е. В. Кислицын, В. С. Рубежанская // e-FORUM. – 2022. – Т. 6, № 1(18). – С. 6. – EDN QTUEDF.