5. Abowd G., Mynatt E. Charting past, present, and future research in ubiquitous computing // ACM Trans. Comput.-Hum. Interact. 2000. № 7(1). P. 29-58.

6. Koutrika G., Ioannidis Y. Personalizing queries based on networks of composite preferences // ACM Trans. Database Syst. 2010. № 35(2). P. 13:1-13:50.

7. Kientz J. Embedded capture and access: encouraging recording and reviewing of data in the caregiving domain // Personal Ubiquitous Comput. 2012. № 16(2). P. 209-221. EDN: OAQAUG.

ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ РЕАЛИЗАЦИИ МОДУЛЬНЫХ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Иваненко О. А., Букреев Д. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: dmitriy.bukreev@mel-su.ru

Научный руководитель: Букреев Д. А.

Введение

Рост функциональной сложности цифровых продуктов и постоянное расширение пользовательских сценариев закономерно приводят к увеличению требований к архитектуре интерфейса. Сегодня пользовательский интерфейс перестает быть простой визуальной надстройкой над программной логикой и становится самостоятельным слоем, отвечающим за качество взаимодействия, удобство восприятия и эмоциональное восприятие продукта. В таких условиях особенно актуальными становятся подходы, позволяющие обеспечить гибкость, структурированность и масштабируемость интерфейсных решений. Одним из них является модульная архитектура пользовательских интерфейсов, которая предполагает построение UI как системы взаимосвязанных, но независимых компонентов.

Интерес к модульным интерфейсам объясняется несколькими факторами. Во-первых, современные цифровые продукты развиваются итеративно, а значит, архитектура должна быть устойчивой к изменению функциональности. Во-вторых, многоплатформенная среда – включая мобильные устройства, планшеты, веб-клиенты – требует единообразия и повторного использования интерфейсных элементов. В-третьих, рост числа технологий и инструментов разработки создаёт возможности для гибкого проектирования интерфейсов, но одновременно повышает требования к их согласованности. Модульный подход позволяет решить эти задачи, обеспечивая независимость компонентов, их композицию и возможность многократного использования в различных контекстах.

Особую значимость модульные интерфейсы приобретают в профессиональных и образовательных цифровых системах, где интерфейс должен быть не только удобным, но и предсказуемым, адаптивным и устойчивым к обновлениям. В таких проектах от интерфейса требуется соблюдение дизайн-системы, согласованность визуальных элементов и отсутствие противоречий в пользовательских сценариях. Именно это делает модульность не просто техническим приёмом, а архитектурной стратегией, влияющей на качество продукта в целом.

Цель исследования – анализ современных технологий и инструментов, применяемых для реализации модульных пользовательских интерфейсов, а также изучение специфики построения таких интерфейсов в условиях разработки образовательных мобильных приложений.

Материалы и методы исследования

Материалом исследования послужили современные технологии проектирования и реализации пользовательских интерфейсов, применяемые в мобильных и кроссплатформенных приложениях. В качестве эмпирической основы использованы практические решения, включая архитектурную модель приложения, структуру модульных компонентов, особенности их взаимодействия и принципы построения интерфейсной части проекта. Дополнительно анализ опирался на документацию инструментов Jetpack Compose, Flutter, React Native и SwiftUI, а также на рекомендации по структуре дизайн-систем и компонентных библиотек, которые определяют актуальные стандарты в сфере UI-разработки.

Методологически исследование основано на сочетании теоретического анализа и сравнительного подхода. На первом этапе были изучены концептуальные источники, посвящённые компонентному проектированию, архитектурным шаблонам пользовательских интерфейсов (MVC, MVP, MVVM) и принципам модульности в интерфейсных системах. Это позволило сформировать целостное представление о фундаментальных подходах, лежащих в основе модульных UI.

На втором этапе проведён сравнительный анализ технологий, применяемых для разработки модульных интерфейсов. Сравнение осуществлялось по ряду критериев: гибкость архитектуры, поддержка композиции компонентов, удобство прототипирования, совместимость с различными платформами, требования к инфраструктуре и возможности расширения функциональности. Такой подход позволил выделить особенности нативных и кроссплатформенных инструментов, а также определить их области рационального применения.

Результаты исследования и их обсуждение

Переход к модульным пользовательским интерфейсам стал естественным ответом на усложнение цифровых систем и рост требований к качеству взаимодействия с пользователем. Современные программы – особенно мобильные и кроссплатформенные – уже не состоят из од-

ного монолитного интерфейса. Они представляют собой совокупность взаимосвязанных компонентов, которые должны быть независимыми, гибкими, легко комбинируемыми и при этом визуально согласованными. Именно эти требования формируют теоретическую основу модульности, позволяя воспринимать интерфейс как динамическую структуру, способную развиваться параллельно с функциональностью приложения. Модульный пользовательский интерфейс можно определить, как систему, построенную из самостоятельных компонентов, которые обладают чётко определённой функциональностью и могут использоваться повторно в различных частях приложения. Такой подход исходит из принципа композиции: интерфейс формируется не из единой монолитной структуры, а из наборов элементов, каждый из которых отвечает за свой участок логики и визуального представления [2].

Важным преимуществом модульного UI является возможность изменять отдельные элементы без воздействия на всю систему. Это существенно повышает устойчивость интерфейса к обновлениям и сокращает трудоёмкость разработки. Модульность также способствует упорядоченности кода: разработчики работают с изолированными блоками, что облегчает тестирование, документирование и расширение функционала.

Такой подход особенно востребован в образовательных мобильных приложениях, где множество интерфейсных элементов – карточки материалов, блоки тестов, интерактивные панели – могут повторяться в различных разделах, оставаясь при этом визуально согласованными.

Компонентное проектирование базируется на нескольких ключевых принципах, определяющих структуру и поведение модульных интерфейсов.

1. Разделение ответственности.
2. Компонуемость.
3. Повторное использование.
4. Независимость и слабое зацепление.
5. Единый визуальный стиль.

Такая организация делает интерфейс одновременно гибким и устойчивым – качеством, необходимым в продуктах, ориентированных на длительное развитие.

Архитектурные модели определяют, каким образом разделяются данные, логика и представление [1]. Для модульных UI наиболее распространены MVC, MVP и MVVM.

1. MVC (Model–View–Controller) – эта модель является одной из первых попыток разделить логику и представление. Контроллер выступает посредником, управляя данными и изменениями UI. Однако в современных приложениях MVC часто оказывается слишком тесно связанной конструкцией.

2. MVP (Model–View–Presenter) – эта архитектура усиливает разделение обязанностей: презентер полностью управляет взаимодействием между моделью и представлением. Модель получает чёткие границы ответственности, а UI становится «тонким» слоем. MVP хорошо подходит для небольших интерфейсов, но может усложнять структуру при росте приложения.

3. MVVM (Model–View–ViewModel) – двусторонняя привязка данных, позволяющая UI автоматически реагировать на изменение состояния модели. Компонент ViewModel отделяет бизнес-логику от визуального слоя, снижая связность элементов и упрощая тестирование. MVVM особенно эффективно работает при создании модульных интерфейсов в таких инструментах, как Jetpack Compose и Flutter, где декларативная модель представления organically сочетается с реактивным управлением состоянием.

Современные технологии и инструменты реализации модульных UI

Современная практика разработки пользовательских интерфейсов характеризуется появлением большого числа инструментов, ориентированных на построение интерфейса как системы взаимосвязанных модулей. Эти инструменты предлагают разные модели взаимодействия, подходы к управлению состоянием и механизмы визуального представления данных, но объединяет их одно: стремление обеспечить декларативный, устойчивый и легко расширяемый UI. Наибольшую значимость при этом приобретают нативные фреймворки нового поколения и кроссплатформенные решения, позволяющие создавать модульные компоненты, которые можно использовать на разных этапах и в разных частях приложения [3].

Появление Jetpack Compose и SwiftUI стало переломным моментом в нативной мобильной разработке. Эти технологии представляют декларативную модель построения интерфейсов, что полностью соответствует идее модульности: разработчик описывает, каким должен быть компонент, а фреймворк самостоятельно управляет его состоянием, деревом элементов и процессом перерисовки.

Jetpack Compose предлагает модель интерфейса, построенную на функциях, которые описывают UI-компоненты. Каждый компонент становится изолированным модулем, который может быть использован в любом месте приложения. Благодаря механизму recomposition интерфейс автоматически обновляется при изменении состояния, а использование ViewModel обеспечивает слабую связность между слоями.

Сильные стороны Compose:

– минимизация шаблонного кода;

– естественная поддержка MVVM;

– удобство создания комплексных UI-элементов;

– гибкость композиции.

SwiftUI – аналогичная декларативная технология для iOS. Она использует структурный подход, где интерфейс описывается через комбинацию структурных элементов.

Преимущества SwiftUI:

– простой способ описания интерфейса;

– высокая согласованность внешнего вида;

– глубокая интеграция с экосистемой Apple;

– возможность комбинирования модулей в иерархию.

Обе технологии создают среду, где модульность – не дополнительная опция, а встроенный принцип, что делает их идеальными для разработки современных интерфейсов [4].

Кроссплатформенные решения позволяют создавать единый UI-код, который работает на разных платформах – Android, iOS, Web, Desktop. В контексте модульных интерфейсов такие технологии оказываются особенно важными, поскольку позволяют переиспользовать компоненты между проектами и платформами.

Flutter предлагает декларативную модель построения интерфейса и использует виджеты как основную единицу UI. Виджет в Flutter по своей сути является модулем: он обладает собственным состоянием и может быть встроен в другие компоненты. Благодаря этому UI легко структурируется в дерево виджетов, что обеспечивает высокую гибкость композиции.

Особенности Flutter, важные для модульности:

– единая система виджетов;

– чёткое разделение Stateless и Stateful компонентов;

– широкие возможности кастомизации;

– высокая согласованность UI на разных платформах.

React Native применяет модель React – компонентный подход, построенный на JavaScript и JSX. Каждый компонент представляет собой отдельный модуль, который может быть многократно использован в разных экранах. Это совпадает с принципами Atomic Design и позволяет создавать структуры высокой сложности.

Kivy менее распространён, он активно используется в образовательных и экспериментальных проектах. Его компонентная архитектура и механизмы разметки позволяют строить гибкие интерфейсы, где модульность поддерживается за счёт файловой структуры и чёткого разделения логики и представления. Кроссплатформенные технологии усиливают идею модульности, позволяя уменьшить дублирование кода и поддерживать единый стиль взаимодействия на разных устройствах [5]. Проектирование модульных интерфейсов не ограничивается только кодом. На этапе концептуальной разработки важную роль играют инструменты прототипирования и дизайн-системы. Figma, Adobe XD и другие инструменты позволяют:

– создавать интерфейсы в виде архитектуры компонентов;

– выделять повторяющиеся элементы в отдельные UI-компоненты;

– управлять стилями, сетками, типографикой;

– формировать дизайн-системы, используемые в кодовой базе.

Дизайн-система становится фундаментом модульности: она определяет правила построения интерфейсов, внешний вид компонентов, их состояние и взаимодействие. На основе дизайн-системы разрабатываются компонентные библиотеки, которые затем используются как в прототипировании, так и в программной реализации. Таким образом, связь между дизайнерскими и техническими инструментами обеспечивает полноценный цикл создания модульного UI – от визуального моделирования до интеграции в кодовую архитектуру.

Особенности реализации модульных UI в образовательных цифровых продуктах

Образовательные мобильные приложения предъявляют особые требования к пользовательскому интерфейсу. Он должен быть понятным для широкого круга пользователей, адаптивным, устойчивым к изменениям и способным поддерживать многокомпонентную структуру содержания. В таких условиях модульный UI становится не просто удобным архитектурным подходом, а фактически необходимым инструментом, обеспечивающим длительное развитие продукта и сохранение качества взаимодействия.

Особенность образовательных систем заключается в том, что интерфейс должен не только отображать информацию, но и поддерживать активное обучение: вовлекать пользователя, обеспечивать последовательность этапов, визуализировать прогресс и соответствовать методическим задачам. Всё это требует использования гибких интерфейсных конструкций, которые легко адаптируются к содержанию и различным контекстам использования.

В отличие от коммерческих приложений, где дизайн может быть ориентирован на маркетинговые задачи, образовательные продукты предъявляют более строгие требования к функциональной ясности и структурированности.

Среди таких требований можно выделить:

1. Понятность и когнитивная доступность.
2. Предсказуемость взаимодействия.
3. Адаптивность к содержанию.
4. Поддержка динамики обучения.
5. Устойчивость и масштабируемость.

Образовательные платформы часто расширяются: добавляются курсы, модули, уровни сложности. Это делает модульный подход единственно рациональным. Таким образом, интерфейс образовательного приложения должен быть не только эстетичным, но и функционально надёжным, что возможно только при грамотной модульной организации UI.

В образовательных приложениях используются устойчивые паттерны интерфейса, которые зарекомендовали себя как эффективные для восприятия и навигации. Среди них:

1. Карточные компоненты, применяемые для представления единиц контента – уроков, заданий, тем.

2. Интерактивные блоки, управляющие ответами, выбором вариантов, переходом по шагам.

3. Вспомогательные панели прогресса, визуализирующие результаты обучения и стимулирующие мотивацию.

4. Модульные списочные структуры, адаптирующиеся под различный объём данных.

5. Диалоговые окна и подсказки, которые помогают ориентироваться в интерфейсе и выполнять задания.

Все эти паттерны отлично сочетаются с модульной архитектурой, поскольку каждый представляет собой отдельный компонент с собственным состоянием и поведением. Их можно многократно использовать на разных экранах и в разных сценариях [7].

Каждый компонент реализован таким образом, чтобы его можно было использовать в других частях интерфейса, модифицировать или расширять без переписывания логики приложения. Именно такая гибкость делает модульный UI особенно ценным в образовательных цифровых продуктах.

Практическая реализация модульного пользовательского интерфейса предполагает не просто применение отдельных компонентов, а осознанное построение архитектуры, в которой каждый элемент занимает своё место и взаимодействует с другими через чётко определённые механизмы [6]. В условиях реальных проектов модульность проявляется не только в структуре отдельных экранов, но и в общей системе управления состоянием, способах интеграции и принципах поддержки дизайн-системы. Именно эти аспекты определяют жизнеспособность интерфейса при изменении требований, масштабировании или добавлении новых функциональных блоков.

Модульная архитектура позволяет организовать разработку таким образом, чтобы каждая часть интерфейса могла развиваться независимо, при этом оставаясь частью целостной системы. На практике такой подход в значительной степени снижает связанность кода, улучшает тестируемость компонентов и обеспечивает единообразие визуальной среды.

Одним из наиболее эффективных способов структурирования модульных интерфейсов является сочетание архитектуры MVVM и концепции Atomic Design. Обе методики дополняют друг друга, обеспечивая как структурную ясность, так и визуальную последовательность.

MVVM обеспечивает организацию логики интерфейса через разделение модели, представления и слоя ViewModel. В рамках модульного UI это особенно важно:

– каждый компонент получает собственную ViewModel или часть общей модели;

– состояние компонента изолировано и не влияет на работу других модулей;

– обновления происходят автоматически благодаря реактивной модели данных.

Это делает интерфейс устойчивым к модификациям и значительно облегчает его тестирование. Выбор конкретного механизма зависит от сложности интерфейса, уровня вложенности модулей и требований к реактивности. Во всех случаях цель остаётся неизменной: обеспечить согласованность модулей при сохранении их автономности.

Заключение

Проведённое исследование показало, что модульные пользовательские интерфейсы становятся ключевым направлением развития современных цифровых систем, обеспечивая структурированность, гибкость и устойчивость интерфейсной архитектуры. Применение компонентного подхода позволяет создавать интерфейсы, которые легко адаптируются к расширению функциональных возможностей, поддерживают единообразие визуального стиля и остаются удобными для сопровождения на протяжении всего жизненного цикла приложения. Анализ теоретических основ модульности продемонстрировал, что архитектурные модели MVC, MVP и особенно MVVM обеспечивают надёжный механизм разделения ответственности между слоями интерфейса. В сочетании с концепцией Atomic Design они формируют фундамент для построения UI-систем, ориентированных на повторное использование и независимость компонентов. Такое объединение архитектурного и визуального подходов позволяет упорядочивать структуру интерфейса и увеличивать его предсказуемость. Рассмотрение современных инструментов – Jetpack Compose, SwiftUI, Flutter, React Native – показало, что именно декларативные технологии предоставляют наиболее гармоничную среду для реализации модульных интерфейсов. Эти фреймворки позволяют разработчику сосредоточиться на логике и назначении компонентов, а управление состоянием и рендерингом берут на себя встроенные механизмы платформы. Дополнение разработки инструментами прототипирования, включая Figma и другие дизайн-системы, обеспечивает согласованность визуального языка и облегчает коммуникацию между разработчиками и дизайнерами.

Список литературы

1. Букреев Д. А., Барановская В. С. Персонализация интерактивных цифровых медиа в образовательной среде // Международный студенческий научный вестник. 2023. № 2.

2. Букреев Д. А., Луц С. М. Особенности разработки web-платформ автоматизированного взаимодействия с клиентом // Современные проблемы геометрического моделирования и информационные технологии: материалы II Межрегиональной научно-практической конференции преподавателей и студентов, посвященной 60-летию образования Мелитопольской школы прикладной геометрии (Мелитополь, 28 мая 2024 года). Мелитополь: Мелитопольский государственный университет, 2024. С. 76-85. EDN: BVUZYR.

3. ISO9241. Ergonomics of human-system interaction. URL: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en (дата обращения: 12.12.2025).

4. Lazuardi M. L., Sukoco I. Design Thinking, David Kelley, Tim Brown: Otak Dibalik Penciptaan Aplikasi Gojek // Organum: JurnalSaintifikManajemen Dan Akuntansi. 2019. № 2.

5. Muslim E., Moch B. N., Wilgert Y., Utami F. F., Indriyani D. User interface redesign of e-commerce platform mobile application (Kudo) through user experience evaluation to increase user attraction. IOP Conference Series: Materials Science and Engineering. 2019. № 508(1). P. 012113.

6. Song's M. Z. Design and implementation of a vue. js-based college teaching system. 2019.

7. Sultan M. Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies: A Comparative Analysis // Proc. Future Technologies Conference. 2017. С. 928-936.

ИСПОЛЬЗОВАНИЕ ОНТОЛОГИЙ В ПОСТРОЕНИИ Е-ПРОФИЛЯ СТУДЕНТА

Иванив О. С., Строкань О. В.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: oksanaivaniv45@gmail.com

Научный руководитель: Строкань О. В.

Введение

В условиях стремительной цифровизации объём создаваемых данных растёт настолько быстро, что традиционные подходы к их обработке и анализу теряют эффективность. Ещё в конце XX века исследователи прогнозировали экспоненциальный рост информационных массивов, и в настоящее время этот прогноз реализуется в полной мере: скорость генерации данных ежегодно увеличивается, при этом значительная их часть относится к неструктурированной информации, требующей новых методов интерпретации и осмысления [1].

Данные процессы оказывают непосредственное влияние на рынок труда. Согласно аналитическим обзорам, специалисты, работающие с анализом и интерпретацией данных, входят в число наиболее востребованных профессионалов в IT-сфере [2]. От них ожидается не только владение инструментами обработки информации, но и способность выявлять взаимосвязи, формировать обоснованные выводы и принимать решения на основе комплексного анализа данных [3]. Повышенный спрос на такие компетенции отражается и в уровне заработных плат, который стабильно превышает среднерыночные показатели [4].

Рост объёмов данных затрагивает и систему высшего образования. Так, современный студент должен уметь ориентироваться в цифровой среде, осваивать новые образовательные форматы и формировать индивидуальную образовательную траекторию. В этих условиях особую значимость приобретает задача структурирования, интеграции и аналитического использования данных о ходе обучения и развитии студентов.

Параллельно трансформируются модели взаимодействия между университетами, государством и работодателями. Ответственность за профессиональную траекторию всё в большей степени возлагается на самого обучающегося, тогда как государственные структуры и бизнес заинтересованы в раннем выявлении и поддержке перспективных кадров. Это формирует запрос на инструменты, способные объединять сведения о компетенциях студентов, динамике их обучения и потребностях рынка труда в единую аналитическую систему.

В данной статье рассматривается онтологический подход как методологическая основа построения электронного профиля студента.

Целью работы является анализ сущности онтологий и семантических технологий, обоснование их применения в образовательных цифровых системах, а также выявление возможностей и ограничений использования онтологий при формировании е-профиля студента.

В работе не только теоретически обосновывается применение онтологий в образовательных цифровых системах, но и предлагается обобщённая онтологическая схема структуры е-профиля студента, иллюстрирующая интеграцию разнородных данных в единую семантическую модель.

Материал и методы исследования

Материалами исследования послужили научные труды, посвящённые онтологическому подходу, семантическим технологиям и цифровизации образовательных процессов [5, 6, 9, 10], а также публикации, рассматривающие вопросы формирования электронных профилей студентов [7] и анализа образовательных траекторий. Дополнительно использовались открытые методические и аналитические материалы в сфере высшего образования, отражающие современные требования к оценке образовательных результатов и компетенций обучающихся.

Методологическую основу исследования составляют системный и онтологический подходы, позволяющие рассматривать е-профиль студента как семантически согласованную модель, объединяющую разнородные данные об учебной, научной, проектной и внеучебной деятельности. В рамках системного подхода электронный профиль анализируется как элемент более широкой экосистемы взаимодействия университета, государства и рынка труда, что соответствует логике, изложенной во введении и выводах статьи.

В исследовании применялись методы концептуального и логического анализа, направлен-