И. С. Клезович
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **Е. В. Рафалова**, ст. преподаватель

РАЗРАБОТКА ПРИЛОЖЕНИЙ НА ЯЗЫКЕ JAVA С ИСПОЛЬЗОВАНИЕМ JAVAFX

Платформа JavaFX отличается простотой освоения и набором инструментов, достаточным для создания десктопных приложений.

Для упрощения создания интерфейса программы был использован инструмент Scene Builder. Scene Builder - это визуальный редактор, позволяющий создавать интерфейсы JavaFX без написания кода. Разработчики добавляют элементы интерфейса путем перетаскивания соответствующего элемента в область окна (сцены) приложения, после чего происходит редактирование их свойств и стилизация элементов. Код реализующий макет в формате fxml генерируется автоматически. Полученный файл интегрируется в проект Java, связывая интерфейс с логикой приложения. В приложениях, разработанных с использованием платформы JavaFX реализуется шаблон проектирования «Модель-представление-контроллер» (MVC).

Компоненты шаблона MVC следующие:

– представление – файл fxml, описывающий пользовательский интерфейс;

– контроллер – класс Java, как правило, реализующий интерфейс Initializable, связывающий представление с логикой приложения;

– модель – объекты домена, определенные на стороне Java, доступные представлению через контроллер.

Для хранения данных приложения определяется серверная база данных, а для обеспечения безопасности данных используется клиент-серверная архитектура. Клиентские компьютеры не имеют прямого доступа к базе данных, а получают только необходимые данные по запросу. Данная архитектура позволяет снизить нагрузку на клиентские машины и обеспечивает централизованное хранение информации.

Описанная схема создания приложения позволяет разработать масштабируемое приложение в короткие сроки.

В. А. Ковалёв
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **О. М. Дерюжкова**, канд. физ.-мат. наук, доцент

АРХИТЕКТУРНЫЕ ШАБЛОНЫ

Архитектурный шаблон – это общее и повторяющееся решение часто возникающей проблемы архитектуры приложений в пределах заданного контекста. Архитектурные шаблоны схожи с шаблонами программного дизайна, однако имеют более широкий охват [1].

Обсудим несколько популярных архитектурных шаблонов, их области применения, преимущества и недостатки более подробно.

Модель-Представление-Контроллер (MVC): разделяет систему на три компонента – модель (логика приложения и данные), представление (отображение пользовательского интерфейса) и контроллер (управление взаимодействием между моделью и представлением).

Где используется:

1. Веб-приложения, где необходимо разделение логики приложения, управления пользовательским интерфейсом и данных.

2. Приложения, ориентированные на интерфейс пользователя, такие как приложения для смартфонов.

Плюсы:

1. Разделение ответственностей: разделяет приложение на три основных компонента – модель (логика данных), представление (отображение данных) и контроллер (управление пользовательским вводом), что облегчает сопровождение и расширение кода.

2. Повторное использование кода: хорошо структурированный MVC позволяет повторно использовать компоненты для других частей приложения или проектов.

3. Улучшенная читаемость кода: четкое разделение слоев делает код более понятным и обеспечивает легкость в его поддержке.

Минусы:

1. Избыточность классов: в некоторых случаях может привести к избыточности классов из-за строгого разделения компонентов.

2. Сложность поддержки: неправильное использование может сделать приложение сложным для понимания и поддержки из-за разветвленной структуры.

Клиент-Сервер: разделяет приложение на клиентскую (отвечает за пользовательский интерфейс) и серверную (отвечает за обработку данных и бизнес-логику) части, взаимодействующие посредством сетевых запросов. Есть общие ресурсы и сервисы, к которым нужно обеспечить доступ большого количества распределенных клиентов, и при этом необходимо контролировать доступ или качество обслуживания [2].

Где используется:

1. Веб-приложения, где клиентская часть взаимодействует с сервером для получения данных.

2. Системы, требующие централизованного управления данными.

Плюсы:

1. Масштабируемость: разделяет функциональность между клиентом и сервером, обеспечивая возможность горизонтального масштабирования и обработки большого числа запросов.

2. Более легкое обновление: изменения в логике клиента или сервера не всегда затрагивают другую часть системы, что упрощает процесс обновления.

Минусы:

1. Сложность сетевого взаимодействия: требует более сложной обработки сетевых запросов и управления ошибками при взаимодействии между клиентом и сервером.

2. Увеличенная нагрузка на сервер: серверу может потребоваться обрабатывать больше запросов, что может привести к увеличению нагрузки и необходимости в масштабируемости сервера.

Одиночка (Singleton): это творческий шаблон проектирования, который позволяет гарантировать, что у класса будет только один экземпляр (рисунок 1), предоставляя при этом глобальную точку доступа к этому экземпляру [3].

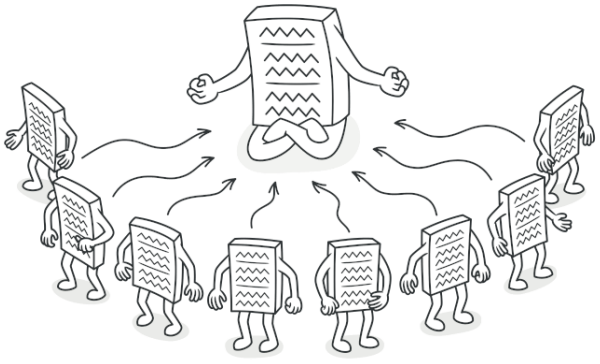

Рисунок 1 – Одиночка (Singleton)

Где используется:

1. Когда необходим только один экземпляр класса в приложении.

2. Для управления общими ресурсами, такими как журналы или настройки.

Плюсы:

1. Гарантированный один экземпляр: обеспечивает наличие только одного экземпляра класса и глобальную точку доступа к этому экземпляру.

2. Управление ресурсами: позволяет эффективнее управлять общими ресурсами.

Минусы:

1. Глобальное состояние: использование глобального объекта может усложнить тестирование и поддержку программы из-за связанности с другими частями кода.

2. Ограничение расширяемости: вносит ограничения на расширение функциональности, потому что только один экземпляр класса может существовать.

Это лишь общий обзор. Часто разработчики комбинируют несколько шаблонов для создания более гибких и масштабируемых систем. Успешное использование архитектурных шаблонов зависит от понимания их преимуществ, недостатков и соответствия требованиям конкретного проекта.

Литература

1. Краткий обзор 10 популярных архитектурных шаблонов приложений [Электронный ресурс]. – 2019. – URL: https://medium.com/nuances-of-programming/краткий-обзор-10-популярных-архитектурных-шаблонов-приложений-81647be5c46f. – Дата доступа:28.01.2024.

2. Самые важные архитектурные шаблоны, которые нужно знать [электронный ресурс]. – 2020. – URL: https://habr.com/ru/companies/alconost/articles/522662/. – Дата доступа: 09.02.2024.

3. Singleton [электронный ресурс]. – 2023. – URL: https://refactoring.guru/design-patterns/singleton. – Дата доступа: 12.02.2024.

М. В. Ковалев, М. А. Гриб, С. Д. Саковский
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **Ю. В. Никитюк**, канд. физ.-мат. наук, доцент

РАЗРАБОТКА СИСТЕМЫ ПОЖАРНОЙ ОХРАНЫ И ДЫМОУДАЛЕНИЯ ДЛЯ МНОГОКВАРТИРНОГО ЖИЛОГО ДОМА

Система пожарной сигнализации – совокупность взаимодействующих технических средств, предназначенных для обнаружения пожара, формирования, сбора, обработки, регистрации и передачи в заданном виде сигналов о пожаре, режимах работы системы, другой информации и выдачи (при необходимости) сигналов на управление техническими средствами противопожарной защиты, технологическим, электротехническим и другим оборудованием [1].

Дымоудаление – это процесс перераспределения воздушных струй в жилых и промышленных зданиях в случае пожара. Дым и другие продукты горения выводятся далеко за пределы помещений, а вместо них поступает чистый воздух.

Система дымоудаления – специализированный комплекс вентиляционного оборудования, предназначенный для оперативного вывода продуктов горения из помещений, освобождения от дыма путей эвакуации людей и способствующий правильной организации мероприятий по устранению возгорания.

Выполняются следующие функции:

– сокращается возможность распространения огня;

– снижается количество дыма;