Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра экономической математики, информатики и статистики (ЭМИС)

MVC МОДЕЛЬ

Реферат по дисциплине «Методы и средства проектирования информационных систем»

Студент гр. 590-1
_____/Г.К. Петров
«___» _____ 2023 г.
Старший преподаватель
кафедры ЭМИС
_____ _____/ Д. П. Вагнер
оценка подпись
«___» _____ 2023 г.

Томск 2023

Оглавление

ВВЕДЕНИЕ

В современном мире программной разработки архитектурный подход MVC (Model-View-Controller) представляет собой ключевой элемент для эффективного построения программного обеспечения. Этот подход является основой многих успешных проектов и фреймворков, обеспечивая высокую степень организации и управляемости кода.

MVC – это сокращение, обозначающее три основных компонента архитектурного стиля: Model (Модель), View (Представление) и Controller (Контроллер). Этот подход предполагает разделение приложения на три логические составляющие, каждая из которых выполняет свою уникальную роль. Модель отвечает за хранение данных и бизнес-логику, Представление — за отображение информации пользователю, а Контроллер — за управление взаимодействием между Моделью и Представлением.

Архитектурный подход MVC обладает высокой степенью структурированности, что облегчает понимание и поддержание кодовой базы. Разделение ответственностей между компонентами позволяет улучшить модульность приложения, делая его более гибким и легким для изменений. Этот подход также способствует повторному использованию кода, поскольку каждый компонент может быть независимо изменен без влияния на остальные.

MVC существенно улучшает сопровождаемость программного обеспечения, поскольку разделение бизнес-логики, представления и управления взаимодействием создает четкие границы между компонентами. Это делает код более предсказуемым, облегчая выявление и исправление ошибок.

Особенно важным аспектом является возможность параллельной разработки компонентов, так как разработчики могут одновременно работать над Моделью, Представлением и Контроллером, минимизируя конфликты и ускоряя процесс разработки. В результате использование архитектурного

подхода MVC становится крайне важным для построения эффективных и масштабируемых программных продуктов.

1 ОСНОВНЫЕ КОМПОНЕНТЫ MVC

1.1 Модель (Model)

Модель в архитектуре MVC представляет собой компонент, отвечающий за управление данными и бизнес-логикой приложения. Ее основная задача – хранить, обрабатывать и предоставлять данные для Представления и Контроллера. Модель не зависит от конкретного способа отображения информации пользователю или способа ее ввода. Она предоставляет интерфейс, с помощью которого другие компоненты могут получать доступ и взаимодействовать с данными.

Модель содержит данные, которые представляют состояние приложения. Эти данные могут включать в себя информацию о пользователях, объектах, операциях и других сущностях, с которыми работает приложение. Кроме того, модель определяет методы и функции для работы с этими данными, включая их изменение, извлечение и фильтрацию.

Примеры реализации модели могут различаться в зависимости от языка программирования. В языке Java моделью может быть класс, инкапсулирующий данные и методы их обработки. В языке Python это может быть объект с атрибутами и методами, обеспечивающими доступ и изменение данных. В веб-разработке на языке JavaScript моделью может быть объект, представляющий структуру данных и содержащий методы для взаимодействия с сервером.

Важно отметить, что конкретная реализация модели может различаться в зависимости от требований конкретного проекта, но основные принципы остаются постоянными – управление данными и обеспечение их доступа для других компонентов системы.

1.2 Представление (View)

Представление в архитектуре MVC ответственно за отображение данных пользователю и предоставление интерфейса для взаимодействия с приложением. Основные задачи и функции представления включают:

- **Отображение данных:** Представление получает данные из Модели и отображает их пользователю в удобной форме. Это может включать в себя отображение текста, графики, таблиц и других элементов интерфейса.
- **Обновление пользовательского интерфейса:** Представление реагирует на изменения данных в Модели и обновляет пользовательский интерфейс соответствующим образом. Это обеспечивает согласованность между данными и их визуальным представлением.
- **Обработка пользовательских событий:** Представление обрабатывает действия пользователя, такие как клики мыши, нажатия клавиш и другие события взаимодействия. Эти события передаются Контроллеру для обработки.

Одним из ключевых принципов архитектуры MVC является отделение отображения от бизнес-логики. Представление не содержит бизнес-логики приложения, а лишь отображает данные, предоставленные Моделью. Это позволяет достичь высокой степени модульности, упрощает тестирование и поддержку кода.

Представление обеспечивает интерфейс для взаимодействия пользователя с приложением. Это включает в себя создание элементов управления (кнопки, поля ввода, списки и т.д.), обработку ввода пользователя и отображение результата визуально. При этом Представление не обрабатывает бизнес-логику, а лишь передает действия пользователя Контроллеру для

соответствующей обработки. Это обеспечивает четкое разграничение ответственностей между компонентами архитектуры MVC.

1.3 Контроллер (Controller)

Контроллер в архитектуре MVC играет решающую роль в управлении взаимодействием между Моделью и Представлением. Его основные задачи включают:

- **Получение пользовательского ввода:** Контроллер отслеживает действия пользователя, такие как клики, ввод текста и другие события, и передает эти события для обработки.
- **Обработка бизнес-логики:** Контроллер содержит логику, связанную с обработкой пользовательских действий. Он взаимодействует с Моделью для получения и обновления данных, а также принимает решения о том, как отреагировать на действия пользователя.
- **Обновление Представления:** Контроллер уведомляет Представление о необходимости обновления, когда происходят изменения в данных. Это обеспечивает согласованность между Моделью и Представлением.

Контроллер отвечает за управление потоком данных и событий в приложении. Он определяет, какие действия предпринять при определенных событиях, какие данные передать в Модель для обработки, и как обновить Представление после изменений. Это обеспечивает эффективное взаимодействие между компонентами архитектуры MVC.

2 ПРИМЕНЕНИЕ MVC В РАЗЛИЧНЫХ ОБЛАСТЯХ

2.1 Веб-разработка

В веб-разработке архитектурный подход MVC широко применяется для построения структурированных и легко поддерживаемых веб-приложений. Фреймворки веб-приложений предоставляют реализацию этой архитектуры, облегчая разработку и поддержку веб-проектов. Примеры таких фреймворков включают Django, Flask (Python), Ruby on Rails (Ruby), Spring MVC (Java), Laravel (PHP), ASP.NET MVC (C#), и другие.

Примеры успешных веб-проектов, использующих архитектуру MVC

1. **Ruby on Rails (Twitter):** Одним из первых успешных веб-проектов, использующих Ruby on Rails, был Twitter. Фреймворк предоставил эффективный механизм обработки запросов, взаимодействия с базой данных и управления пользовательским интерфейсом, что способствовало быстрому развитию платформы.
2. **Django (Instagram):** Instagram, одно из самых популярных приложений для обмена фотографиями, построено с использованием фреймворка Django. Django обеспечил структурированность кода, удобное взаимодействие с базой данных и возможность масштабирования, что сделало платформу успешной и масштабируемой.
3. **Laravel (Wikipedia):** Веб-версия Википедии использует фреймворк Laravel. Laravel предоставляет удобный синтаксис для работы с базой данных, маршрутизацией и представлениями, что делает разработку и поддержку многопользовательской платформы более простой и эффективной.

Применение архитектуры MVC в веб-разработке не только упрощает разработку, но также обеспечивает четкое разделение ответственностей между

компонентами, что является ключевым фактором в поддержке и масштабировании веб-приложений.

2.2Десктопные приложения

Применение архитектурного подхода MVC в десктопных приложениях обеспечивает эффективное разделение бизнес-логики, пользовательского интерфейса и управления событиями.

Примеры популярных десктопных приложений, использующих MVC

1. **Microsoft Excel:** Этот популярный офисный инструмент использует архитектуру MVC для обработки данных (ячейки, формулы) в Модели, отображения этих данных в таблицах и графиках через Представление, а также обработки действий пользователя через Контроллер.
2. **Adobe Photoshop:** Графический редактор Photoshop применяет архитектуру MVC для эффективной работы с изображениями. Модель обрабатывает данные изображения, Представление отображает их на экране, а Контроллер обрабатывает действия пользователя, такие как рисование или изменение параметров.
3. **Eclipse IDE:** Интегрированная среда разработки Eclipse использует MVC для управления проектами, редактирования кода и визуализации результатов. Это обеспечивает четкую структуру при работе с большими кодовыми базами.

Применение архитектуры MVC в десктопных приложениях повышает читаемость кода, упрощает его сопровождение и обеспечивает более гибкую архитектуру, что особенно важно при разработке сложных и функциональных приложений.

2.3 Мобильные приложения

Применение архитектуры MVC в мобильных приложениях обеспечивает легкость разработки, тестирования и поддержки, что является ключевым фактором успеха для многих популярных приложений.

1. **Instagram (iOS и Android):** Instagram, одно из самых популярных мобильных приложений для обмена фотографиями, использует архитектуру MVC. Модель обрабатывает данные, такие как изображения и пользовательские активности, Представление создает графический интерфейс, а Контроллер управляет навигацией и обработкой событий.
2. **Evernote (iOS и Android):** Evernote, приложение для создания и хранения заметок, также использует MVC. Модель управляет данными (заметками), Представление создает интерфейс для их отображения, а Контроллер обрабатывает действия пользователя и управляет переходами между различными экранами.
3. **Google Maps (iOS и Android):** Приложение Google Maps, предоставляющее картографические данные и навигацию, применяет архитектуру MVC. Модель управляет геоданными, Представление создает интерфейс для взаимодействия с картой, а Контроллер обрабатывает действия пользователя, такие как увеличение, перемещение и поиск мест.

ЗАКЛЮЧЕНИЕ

Архитектурный подход MVC (Model-View-Controller) представляет собой мощный инструмент в разработке программного обеспечения, который успешно применяется в различных областях, включая веб-разработку, десктопные приложения и мобильные приложения. Разделение приложения на три основных компонента – Модель, Представление и Контроллер – обеспечивает четкое разделение ответственностей и способствует построению гибких, масштабируемых и легко поддерживаемых систем.

В веб-разработке, MVC служит фундаментальной основой для построения веб-приложений, обеспечивая разделение логики бизнеса, представления данных и обработки взаимодействия с пользователем. Для десктопных приложений, архитектура MVC обеспечивает структурированное управление графическим пользовательским интерфейсом и логикой приложения. В мобильной разработке, где требования к адаптивности и интерактивности высоки, MVC способствует эффективной организации кода и взаимодействия компонентов.

Опыт успешных проектов, таких как Instagram, Google Maps и другие, подтверждает применимость и эффективность архитектуры MVC. Однако важно учитывать, что выбор архитектуры зависит от конкретных требований проекта, и в некоторых случаях, возможно, более современные подходы, такие как MVVM (Model-View-ViewModel), MVP (Model-View-Presenter), и др., могут быть более подходящими.

В целом, архитектурный подход MVC продолжает оставаться актуальным и востребованным в мире разработки программного обеспечения, предоставляя надежный фреймворк для построения современных и эффективных приложений.

Список использованных источников

1. "Шаблоны проектирования" Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. В этой книге рассматриваются шаблоны проектирования, включая MVC.

2. MVC (Model-View-Controller): что это такое [Электронный ресурс]: сайт SkillFactory. URL: https://blog.skillfactory.ru/glossary/mvc/ (дата обращения: 18.12.2023) .

3. MVC для веб: проще некуда / Хабр. [Электронный ресурс]: сайт производителя программных решений для управления бизнес-процессами SAP. URL: https://habr.com/ru/articles/181772/ (дата обращения: 18.12.2023) .

4. Что такое MVC: рассказываем простыми словами [Электронный ресурс]: Хекслет. URL: https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami (дата обращения: 17.12.2023).