|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА** | | |
| **ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА**  бул. Пещерско шосе № 26  4002 гр. Пловдив, България  тел. 032 / 643-675  info-1690174@edu.mon.bg | A logo with a black background  Description automatically generated | **VOCATIONAL SCHOOL OF ELECTRICAL ENGINEERING AND ELECTRONICS**  26 Peshtersko chaussee blvd.,  4002 Plovdiv, Bulgaria  Phone: 032 / 643-675  info-1690174@edu.mon.bg |

**професия код 481030 „Приложен програмист“**

**специалност код 4810301 „Приложно програмиране“**

**ДОКЛАД**

**Тема:** Архитектурен шаблон MVC (Model-View-Controller)

**Разработил**: Богомил Даниелов Петров №4

Клас: 11. А

e-mail: [bogomilpetrov21a@gmail.com](mailto:bogomilpetrov21a@gmail.com)

**Ръководител-консултант**: инж. Ваня Менгова

Пловдив

2025

**СЪДЪРЖАНИЕ**

[**УВОД 2**](#_Toc181002129)

[**ОСНОВНА ЧАСТ 3**](#_Toc181002130)

[**ГЛАВА 1 - ИСТОРИЯ И ЕВОЛЮЦИЯ НА MVC 3**](#_Toc181002131)

[**ГЛАВА 2 - ОСНОВНИ КОМПОНЕНТИ НА MVC 4**](#_Toc181002132)

[**ГЛАВА 3 - ПРИНЦИПИ НА РАБОТА НА MVC 5**](#_Toc181002133)

[**ГЛАВА 4 - ПРЕДИМСТВА И НЕДОСТАТЪЦИ НА MVC 6**](#_Toc181002134)

[**ГЛАВА 5 - ПРИЛОЖЕНИЕ НА MVC 7**](#_Toc181002135)

[**ГЛАВА 6 - АЛТЕРНАТИВИ НА MVC 8**](#_Toc181002136)

[**ЗАКЛЮЧЕНИЕ 9**](#_Toc181002137)

[**ИЗТОЧНИЦИ 10**](#_Toc181002138)

[**ПРИЛОЖЕНИЯ 11**](#_Toc181002139)

# УВОД

Съвременните софтуерни архитектури разчитат на структуриран подход за организация на кода, което позволява по-лесна поддръжка, гъвкавост и разширяемост на приложенията.

Архитектурният шаблон MVC е една от водещите архитектурни структури, използвана широко заради разделението на основните компоненти на приложението. Това разделение прави шаблона особено подходящ за сложни и мащабни проекти, където ясното разделение на отговорности улеснява както разработчиците, така и бъдещата поддръжка на системата.

MVC е популярен заради своята модулност и структура, което го прави адаптивен към разнообразни софтуерни решения. Чрез разделение на функциите по роли, този шаблон осигурява контрол върху сложността на софтуера и допринася за по-добро организиране на работния процес.

# ОСНОВНА ЧАСТ

## ГЛАВА 1 - ИСТОРИЯ И ЕВОЛЮЦИЯ НА MVC

***1.1. История***

Архитектурният шаблон MVC е създаден през 70-те години на миналия век като част от програмната парадигма на Smalltalk, разработена в Xerox PARC.([3.](#_ИЗТОЧНИЦИ)) Идеята е да се улесни взаимодействието между потребителя и приложението, като се осигури лесен начин за управление на големи обеми данни и сложни изчисления. Основната мотивация зад създаването на MVC е осигуряването на структура, която разделя данните от представянето и логиката, като същевременно осигурява лесен достъп до всеки компонент за разработчиците.

***1.2. Еволюция***

В началото MVC е използван основно за настолни приложения, но с нарастващата сложност на уеб приложенията през 90-те и 2000-те години, модела се адаптира, ставайки популярен за уеб приложения. Това довежда до появата на различни варианти на MVC, като MVVM (Model-View-ViewModel) и MVP (Model-View-Presenter), които се стремят към още по-добра разделеност на компонентите и опростена работа с потребителския интерфейс. Въпреки промените, основните принципи на MVC остават същите и продължават да играят ключова роля в съвременното програмиране, особено при разработка на интерфейси и клиент-сървър приложения.

## ГЛАВА 2 - ОСНОВНИ КОМПОНЕНТИ НА MVC

***2.1. Model (Модел)***: Моделът е компонентът, който се грижи за управлението на данните в системата. Той отговаря за запазването, промяната и извличането на данни, които са необходими за логиката на приложението. Моделът често е свързан с базата данни и осигурява интерфейса за достъп до данните, необходими на приложението.

***2.2. View (Изглед)***: Изгледът е компонентът, който отговаря за визуализацията на информацията пред потребителя. Той извежда данните от модела във вид, подходящ за потребителите. Изгледът е единственият компонент в MVC архитектурата, който директно се свързва с потребителя, и съответно отговаря за потребителското изживяване и удобство.

***2.3.*** ***Controller (Контролер)***: Контролерът е междинен компонент, който осъществява връзката между модела и изгледа. Той получава заявките от потребителя, обработва ги и осъществява необходимите промени в модела, след което обновява изгледа. Контролерът е отговорен за управлението на взаимодействието между модела и изгледа, като осигурява логиката и потока на информацията в системата.

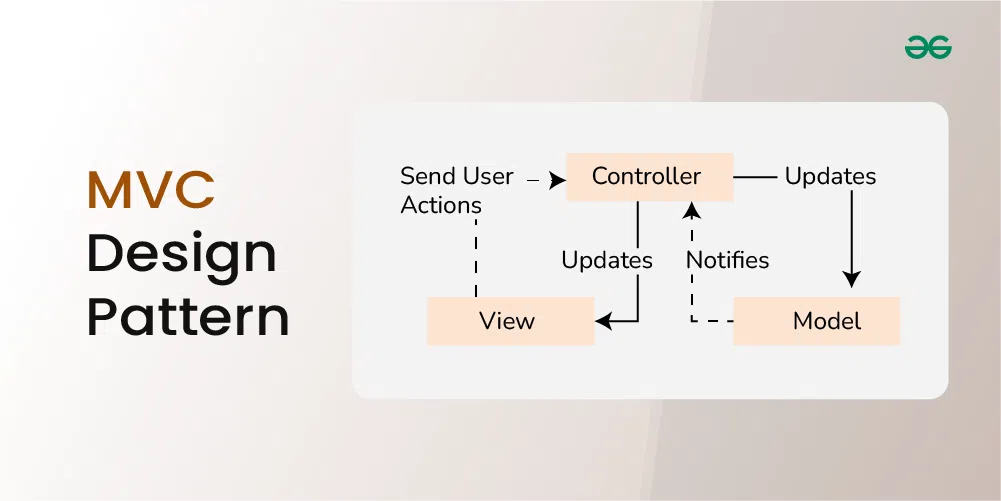
## ГЛАВА 3 - ПРИНЦИПИ НА РАБОТА НА MVC

***3.1. Принципи***

Трите компонента на MVC си взаимодействат по ясно структуриран начин:

1. Потребителят извършва действие в **изгледа**, което изпраща ***заявка*** към ***контролера***.
2. ***Контролерът*** обработва заявката и комуникира с ***модела*** за извличане или обновяване на данни.
3. След като данните са обработени, ***контролерът*** предава резултата на ***изгледа***.
4. ***Изгледът*** визуализира информацията на потребителя.

***3.2. Примерен работен поток***

[](https://media.geeksforgeeks.org/wp-content/uploads/20240219100940/MVC-design-pattern.webp)Потребителят изпраща заявка за достъп до конкретна информация. **Контролерът** я получава, обработва и се обръща към **модела** за извличане на нужните данни. След като **моделът** върне данните, **контролерът** ги предава на **изгледа**, който ги визуализира пред потребителя

## ГЛАВА 4 - ПРЕДИМСТВА И НЕДОСТАТЪЦИ НА MVC

***4.1. Предимства:***

* Модулност – всеки компонент може да бъде разработван и променян независимо;
* Повторно използване на кода – модулността позволява лесно преизползване на логиката;
* Тестване – отделните компоненти могат да бъдат лесно тествани, без да влияят на останалите компоненти.

***4.2. Недостатъци:***

* Сложност – може да бъде сложен за прилагане в малки проекти;
* Многостранно взаимодействие – изисква тясно взаимодействие между компонентите, което може да е трудоемко.

## ГЛАВА 5 - ПРИЛОЖЕНИЕ НА MVC

Архитектурният шаблон Model-View-Controller (MVC) е популярен избор за разработка на различни софтуерни приложения, включително уеб, настолни и мобилни приложения. Неговото приложение обаче има различни особености в зависимост от платформата, на която се използва.

***5.1. В уеб приложенията***

Уеб приложенията обикновено използват MVC като основен архитектурен шаблон, тъй като уеб средата изисква отделяне на логиката и данните от визуализацията, за да се улесни обслужването на много потребители. Много фреймуърци като ASP.NET MVC, Ruby on Rails и Django използват MVC като основа за организация на уеб приложения. В този контекст MVC осигурява следното:

1. Model (Модел) – управлява взаимодействието с базата данни и обработката на данни. Например, в Django моделът се свързва с ORM (обектно-релационното моделиране), което осигурява интерфейс за работа с бази данни без пряко използване на SQL заявки;
2. View (Изглед) – отговаря за HTML/CSS визуализацията. Изгледите получават данните от контролера и ги визуализират в браузъра на потребителя.
3. Controller (Контролер) – получава HTTP заявки от потребителите, обработва ги и решава какви данни да се извлекат и какъв изглед да се покаже.

***5.2. В настолните приложения***

В настолните приложения MVC също намира широко приложение. Настолните приложения често имат сложен потребителски интерфейс и изискват по-добра организация на кода за по-лесна поддръжка и мащабиране. Примери за такива приложения са тези, разработени с JavaFX, Qt или Objective-C/Cocoa за macOS.

***5.3. В мобилните приложения***

В мобилните приложения MVC се използва с цел подобряване на поддръжката и мащабируемостта на приложенията, като се поддържа ясна структура. В iOS, например, MVC е част от дизайна на UIKit и често се използва за разделение на потребителския интерфейс от логиката.

## ГЛАВА 6 - АЛТЕРНАТИВИ НА MVC

Съществуват различни архитектурни шаблони, които са подобни на MVC, но се различават по начина на взаимодействие между компонентите.

Един от тях е MVVM (Model-View-ViewModel), който се използва основно при създаването на приложения с богато потребителско изживяване, като се фокусира върху представянето на данни и взаимодействието с потребителя.

Друг популярен модел е MVP (Model-View-Presenter), който предоставя допълнителен слой за абстракция и улеснява тестовете.

MVC се избира, когато е необходима строго разграничена структура за управление на данни и взаимодействие с потребителя. В зависимост от конкретните нужди на приложението, алтернативите като MVVM и MVP могат да са по-подходящи, особено при по-сложни интерфейси.

# ЗАКЛЮЧЕНИЕ

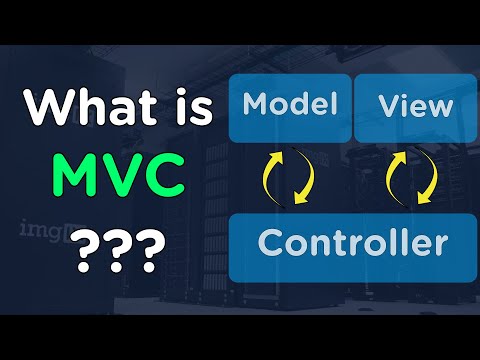
MVC е архитектурен шаблон с доказана ефективност и се използва широко в съвременната софтуерна разработка. Неговите предимства включват ясното разделение на логиката, което улеснява работата на екипите и гарантира по-добра поддръжка и гъвкавост на приложенията. MVC има голямо значение за съвременните уеб и настолни приложения и вероятно ще продължи да бъде популярен шаблон и в бъдеще, поради способността му да адаптира софтуера към бързо развиващите се технологии.

Използването на MVC предоставя ясна и структурирана основа за разработване на мащабни проекти, като същевременно създава стабилност и предвидимост на софтуера. В заключение, MVC остава неизменна част от модерното програмиране, предоставяйки необходимите инструменти за организиране на сложни софтуерни проекти.

# ИЗТОЧНИЦИ

1. Microsoft Docs - MVC Design Pattern, <https://learn.microsoft.com/en-us/aspnet/mvc/overview>, достъпно до 01.11.2024
2. Geeks for Geeks - Model View Controller (MVC) Architecture, <https://www.geeksforgeeks.org/mvc-design-pattern>, достъпно до 01.11.2024
3. Wikipedia - Model–View–Controller, [https://en.wikipedia.org/wiki/Model–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller), достъпно до 01.11.2024
4. MDN Web Docs - Introduction to MVC, <https://developer.mozilla.org/en-US/docs/Glossary/MVC>, достъпно до 01.11.2024
5. Guru99 - What is MVC Architecture?, <https://www.guru99.com/mvc-tutorial.html>, достъпно до 01.11.2024

# ПРИЛОЖЕНИЯ

[](https://www.youtube.com/embed/DUg2SWWK18I?feature=oembed)