# O que é padrão MVC? Entenda arquitetura de softwares!

O MVC é uma sigla do termo em inglês Model (modelo) View (visão) e Controller (Controle) que facilita a troca de informações entre a interface do usuário aos dados no banco, fazendo com que as respostas sejam mais rápidas e dinâmicas.

Você já parou para pensar o que se passa por trás de uma tela de login de um software? Em frações de segundos a página é capaz de absorver as informações que foram digitadas no campo de email e senha, realizar a validação e entregar uma resposta positiva ou negativa. 

Esse processo só se torna possível quando existe um padrão de arquitetura de software adequado. Embora exista vários que podem ser utilizados, o **MVC é o mais conhecido e empregado entre os desenvolvedores profissionais**.

Se você quer saber como [melhorar a usabilidade do seu software](https://endeavor.org.br/estrategia-e-gestao/usabilidade/)e otimizar o tempo de resposta entre o banco de dados e a interface de usuário, continue a leitura que vamos explicar todos os detalhes neste artigo. 

**Afinal, o que é MVC?**

Apesar de muitas pessoas considerarem essa sigla como um padrão de design de interface, na verdade ele **é um padrão de arquitetura de software responsável por contribuir na otimização da velocidade entre as requisições feitas pelo comando dos usuários**. 

Com quase 50 anos de formulação, a arquitetura MVC é dividida em três componentes essenciais: Model, Controller e View. 

Um dúvida muito recorrente na programação é se no processo de desenvolvimento pode ter apenas esses 3 componentes ou se é possível acrescentar mais alguns. A resposta é sim para a possibilidade de inserir uma camada extra. Essa sequência de códigos pode ser alterada conforme a necessidade do projeto. 

**Mas um código com muitas camadas se torna muito confuso e por isso, o ideal é manter o padrão original**. A seguir vamos explicar os conceitos e aplicações dos componentes que acompanham essa arquitetura de software.

**Model ou Modelo**

Essa classe também é conhecida como Business Object Model (objeto modelo de negócio). **Sua responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas**. 

Ele é o detentor dos dados que recebe as informações do Controller, válida se ela está correta ou não e envia a resposta mais adequada. 

**Controller ou Controlador**

**A camada de controle é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model**, processando os dados que o usuário informou e repassando para outras camadas. 

Numa analogia bem simplista, o controller  operaria como o ‘’maestro de uma orquestra’’  que permite a comunicação entre o detentor dos dados e a pessoa com vários questionamentos no MVC. 

**View ou Visão**

**Essa camada é responsável por apresentar as informações de forma visual ao usuário**. Em seu desenvolvimento devem ser aplicados apenas recursos ligados a aparência como mensagens, botões ou telas. 

Seguindo nosso processo de comparação o View está na linha de frente da comunicação com usuário e é responsável transmitir questionamentos ao controller e entregar as respostas obtidas ao usuário. É a parte da interface que se comunica, disponibilizando e capturando todas as informação do usuário.

**Como os componentes interagem?**

Tudo começa com a interação do usuário na camada View. A partir daí o controlador pega essa informações e envia para o Model que fica responsável por avaliar aqueles dados e transmitir uma esposta. 

O controlador recebe essas respostas e envia uma notificação de validação daquela informação para a camada visão, fazendo com a mesma apresente o resultado de maneira gráfica e visual.

Todo esse processo leva em consideração as regras de negócio aplicadas na construção de todo projeto.

**Por que usar MVC?**

Muitos [bootcamps de programação](https://www.lewagon.com/pt-BR/blog/bootcamps-programa%C3%A7%C3%A3o-funcionam) ensinam esse padrão de arquitetura de software por alguns benefícios que justificam o MVC como uma das mais escolhidas no processo de desenvolvimento. Esses benefícios são: 

* **Segurança**: O controller funciona como uma espécie de filtro capaz de impedir que qualquer dado incorreto chegue até a camada modelo.
* **Organização**: Esse método de programação permite que um novo desenvolvedor tenha muito mais facilidade em entender o que foi construído, assim como os erros se tornam mais fácil de serem encontrados e corrigidos.
* **Eficiência**: Como a arquitetura de software é dividida em 3 componentes , sua aplicação fica muito mais leve, permitindo que vários desenvolvedores trabalhem no projeto de forma independente.
* **Tempo**: Com a dinâmica facilitada pela colaboração entre os profissionais de desenvolvimento, o projeto pode ser concluído com muito mais rapidez, tornando o projeto escalável.
* **Transformação**: As mudanças que forem necessárias também são mais fluidas, já que não será essencial trabalhar nas regras de negócio e correção de bugs.

Além disso, um software precisa ter estabilidade no processo de comunicação entre seus elementos de maneira dinâmica para que a experiência do usuário não seja prejudicada. 

**Como implementar um projeto MVC?**

Essa arquitetura de software pode ser utilizada no programação web, mobile ou desktop e ela pode ser implementada através de diversos frameworks de Java como o [Spring MVC](https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#mvc) ou [Play Framework](https://www.playframework.com/) ou também em frameworks mais modernos de Ruby como Ruby on Rails.

Cada um tem suas diferenças, portanto o ideal é que o programador entenda os recursos disponíveis no processo de desenvolvimento e adote o mais adequado a sua necessidade. 

**Conclusão**

O MVC funciona como um padrão de arquitetura de software que melhora a conexão entre as camadas de dados, lógica de negócio e interação com usuário. Através da sua divisão em três componentes, o processo de programação se torna algo mais simples e dinâmico.

Por padrão existem a camada Model, Controller e View que deram origem a sigla dessa arquitetura de software mais utilizado entre os desenvolvedores.  

Algumas empresas podem até cobrar o conhecimento de determinados frameworks para sua aplicação no dia-a-dia, por isso é interessante que o candidato participe de bootcamps que explorem o assunto de maneira adequada, a fim de melhorar a [curva de aprendizado](https://www.lewagon.com/pt-BR/blog/aprender-programacao-do-zero-por-onde-comecar) de quem quer explorar a área.

