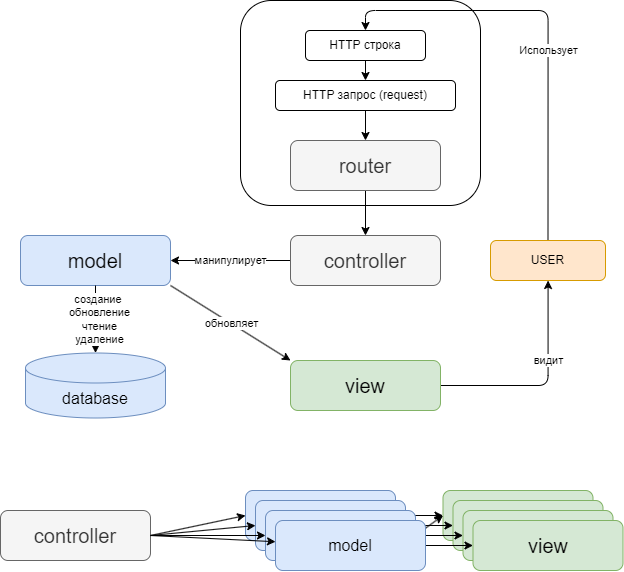


* **Модель** (*Model*) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* **Представление** (*View*) отвечает за отображение данных модели пользователю, реагируя на изменения модели. Это может быть HTML для веб-страниц, JSON для API или даже текст для командной строки
* **Контроллер** (*Controller*) интерпретирует действия пользователя, оповещая модель о необходимости изменений.
* **Роутер** (*Router*) отвечает за определение того, какой контроллер и какое действие должны быть вызваны при определенном запросе



Архитектура MVC (Model-View-Controller) имеет множество преимуществ, но также есть и некоторые недостатки:

1. [Сложность: MVC может быть сложным для понимания, особенно для новичков](https://www.specialist.ru/news/4981). [Это может увеличить время разработки и обучения](https://www.specialist.ru/news/4981).
2. [Связанность: Хотя MVC предполагает разделение ответственности, модель, представление и контроллер часто остаются взаимосвязанными](https://www.specialist.ru/news/4981). [Это может затруднить тестирование и отладку](https://www.specialist.ru/news/4981).
3. [Накладные расходы: MVC может привести к избыточности кода и повышенным накладным расходам, особенно для простых приложений](https://www.specialist.ru/news/4981).
4. [Сложность обновления представления: В некоторых случаях, когда модель изменяется, может быть сложно определить, какие части представления нужно обновить](https://www.specialist.ru/news/4981).