Software Systems Planning (Gpo 101)

TC3004B.101

Actividad 1 - Configuración de ambientes y creación de servlets (Adolfo)

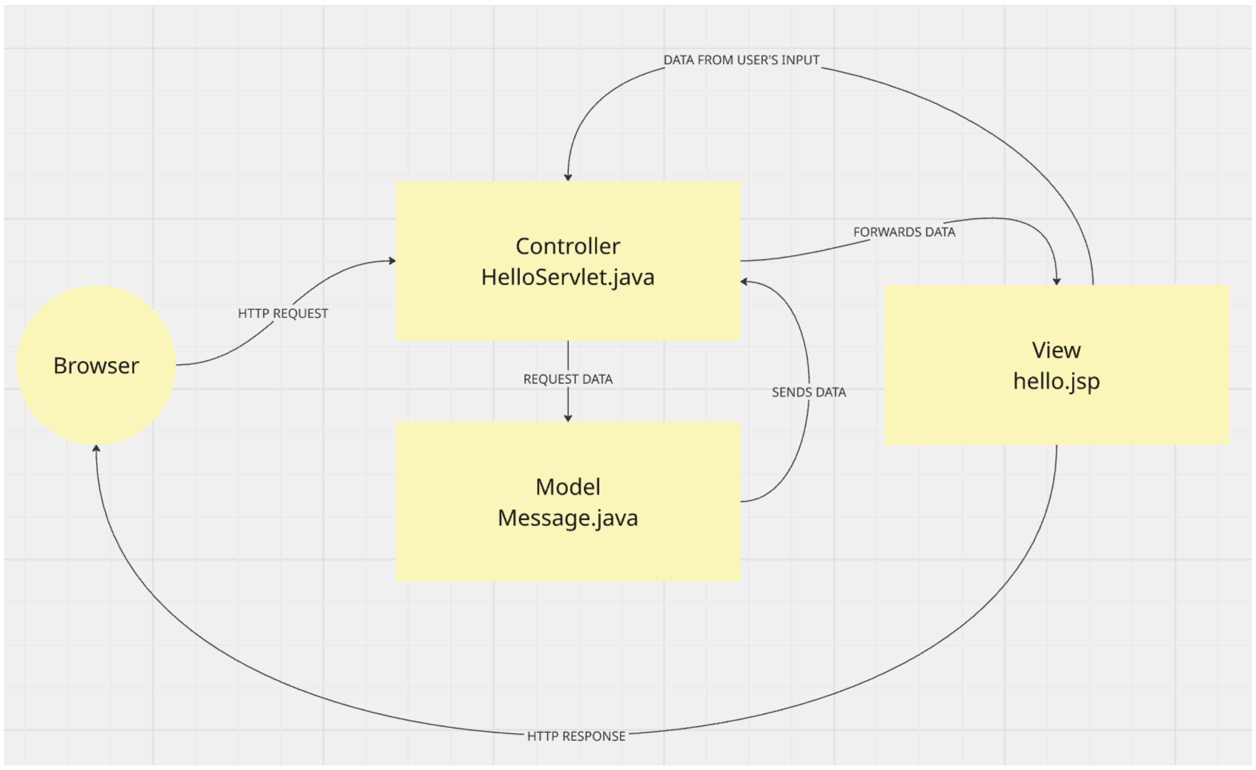
Profesor

Adolfo Arroyo

Diego Iván Rodríguez Núñez
Bachelor of Computer Science
Tecnológico de Monterrey

Fecha:
20 de Febrero de 2025

1. Controller (HelloServlet.java): The brain. It receives the HTTP request from the browser, processes the input, creates or updates the Model, and finally routes the user to the correct View.
2. Model (Message.java): The data. It represents the information being passed around (e.g., a text string, a user profile, a database record).
3. View (hello.jsp): The face. It takes the data from the Model and renders it into HTML for the user to see. It contains zero business logic.



In MVC, the Model manages the core data and business rules, while the View renders this information into a user-friendly format, such as HTML. The Controller acts as the intermediary, processing user requests, updating the Model, and selecting the appropriate View to display. This isolation ensures that developers can modify the interface without disrupting the underlying data structures.

Building upon MVC's presentation-level separation, layered architecture organizes an entire application into horizontal, hierarchical tiers. A standard system includes a Presentation layer for user interactions, a Business Logic layer for executing core rules, and a Data Access layer for database management. Each tier communicates exclusively with the one directly below it, establishing a strict top-down data flow. This structure prevents individual components from becoming overloaded, which promotes code reusability and simplifies testing.

As applications scale, traditional monolithic architectures often transition into microservices. This divides an application into small, autonomous services that communicate through lightweight APIs, with each service focusing on a specific business domain. While a monolith requires redeploying and scaling the entire application for any minor change, microservices allow individual components to be updated and scaled independently. This distributed approach provides targeted scalability and allows engineering teams to use the most efficient technologies for each specific service.