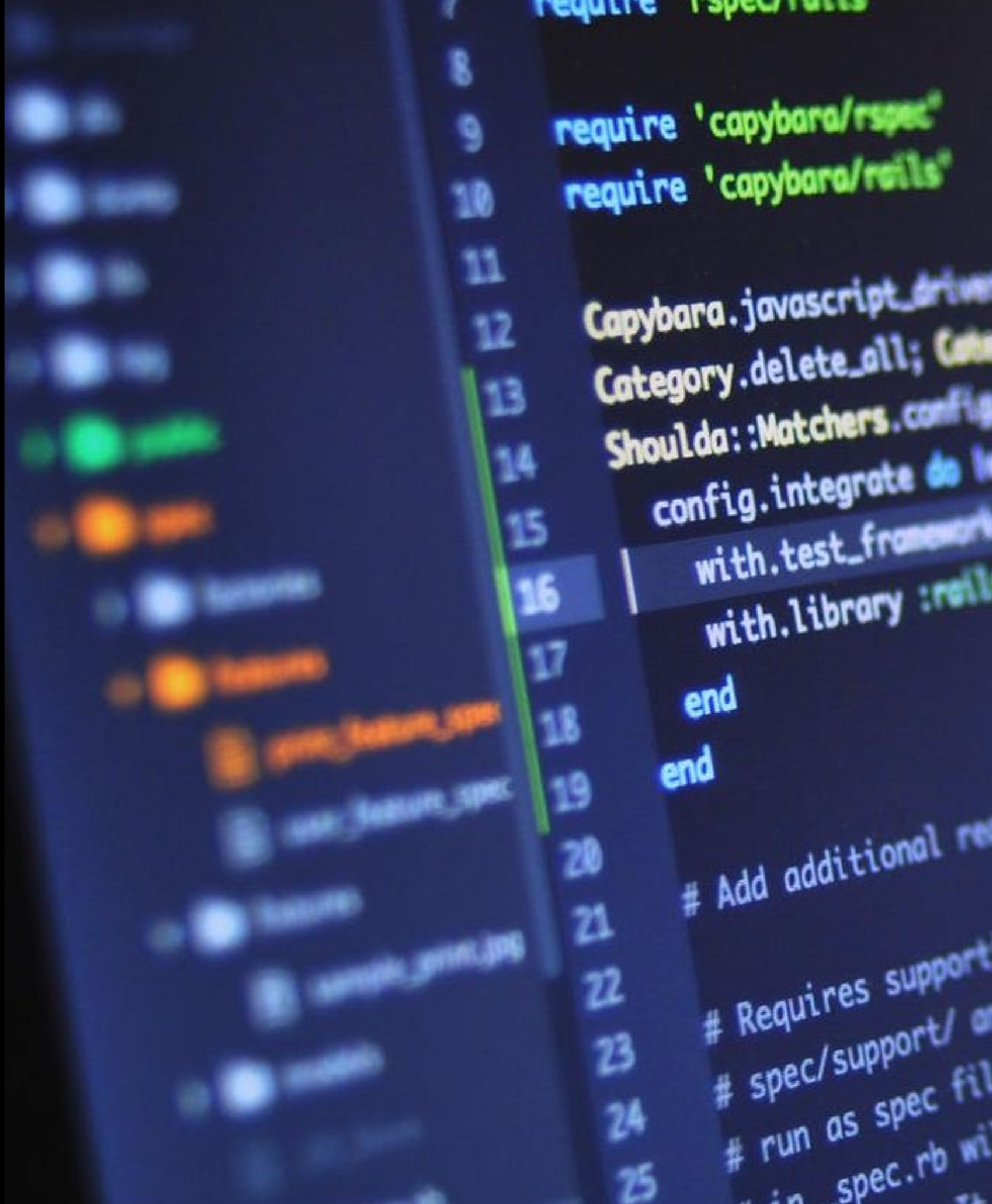
Arquitectura Hexagonal con Java/Quarkus

01-02-03 Y 04 DE JULIO
ESPAÑA - 2024



User Interfaces, Testing, and Case Studies

Esta diapositiva proporciona una descripción general de alto nivel de las interfaces de usuario,

metodologías de prueba y estudios de casos del mundo real.

User Interface Design

Simplicity

Designing clean and uncluttered interfaces that focus on essential functionality and minimize distractions

Usability

Ensuring interfaces are easy to navigate, intuitive, and responsive to user needs and behaviors

Accessibility

Prioritizing inclusive design principles to accommodate users with diverse abilities and needs

By following the principles of simplicity, usability, and accessibility, user interface designers can create seamless and enjoyable experiences that empower all users.

Integrating with the Core

- Primary Adapters (e.g., REST Controllers)

  Use REST controllers or similar primary adapters to connect the user interface with the application core, serving as the entry point for user interactions.

- Encapsulate Business Logic in the Domain

  Ensure that the core business logic is encapsulated within the domain layer, separating it from the user interface and other technical concerns.

- Isolate User Interface from Application Core

  By using the primary adapters, the user interface is isolated from the application core, allowing for easier testing, maintainability, and potential future changes in the underlying architecture.

- Adhere to Separation of Concerns

  The separation of the user interface, primary adapters, and domain layers promotes a clean separation of concerns, improving the overall structure and testability of the application.

-

- Facilitar el desacoplamiento y la flexibilidad

  This approach allows for the user interface and application core to evolve independently, enhancing the overall flexibility and maintainability of the system.

Frameworks and Tools

Angular Integration

Configure Angular to work seamlessly with Quarkus, leveraging Quarkus's reactive programming model and efficient resource management to create a responsive and performant user interface.

React Integration

Integrate React with Quarkus, taking advantage of Quarkus's built-in support for popular web frameworks and its ability to optimize resource utilization, resulting in a smooth and interactive user experience.

Thymeleaf Integration

Integrate Thymeleaf, a popular server-side Java template engine, with Quarkus to build dynamic and intuitive user interfaces, leveraging Quarkus's efficient resource management and rapid application startup capabilities.

Quarkus Development Server

Utilize Quarkus's development server to enable live reloading and hot swapping of your framework-based user interface components, streamlining the development workflow and accelerating the iterative design process.

Quarkus Tooling

Leverage the powerful tooling ecosystem around Quarkus, including IDE integrations and CLI tools, to seamlessly manage the development, configuration, and deployment of your framework-powered user interface applications.

Hexagonal Architecture Testing

Unit Testing Entities

Verifying the individual behavior of entities within the hexagonal architecture, ensuring they adhere to their expected functionality and business rules.

Integration Testing Services

Validating the integrated behavior of services, ensuring they properly communicate and exchange data with the entities and adapters.

Adapter Testing

Ensuring the adapters (ports and adapters) correctly handle the interaction between the application and external systems or user interfaces.

End-to-End Testing

Verifying the overall functionality of the application by simulating real-world scenarios and validating the end-to-end flow through the hexagonal architecture.

By thoroughly testing the individual entities, integrated services, and adapters, you can ensure the correct functioning of the application within the hexagonal architecture, leading to a more reliable and maintainable system.

Unit Testing

Unit tests are crucial for ensuring the reliability and maintainability of software systems. In the context of a web application, unit tests can be used to validate the behavior of core services, such as the Order Service, and the corresponding REST controllers, like the Order Resource. These tests ensure that individual components function as expected, regardless of the system's overall complexity.

Integration Testing

- **User Interface and Core Integration**

  Verifying that the user interface seamlessly integrates with the core functionality of the application, ensuring a cohesive and responsive user experience.

- **Primary Adapter Validation**

  Validating the primary adapters that facilitate communication between the user interface and the core, such as API calls, data transformations, and event handling.

- **End-to-End Workflow Testing**

  Simulating real-world user scenarios and verifying that the entire workflow, from user input to core processing and back, functions as expected.

- **Input Validation and Error Handling**

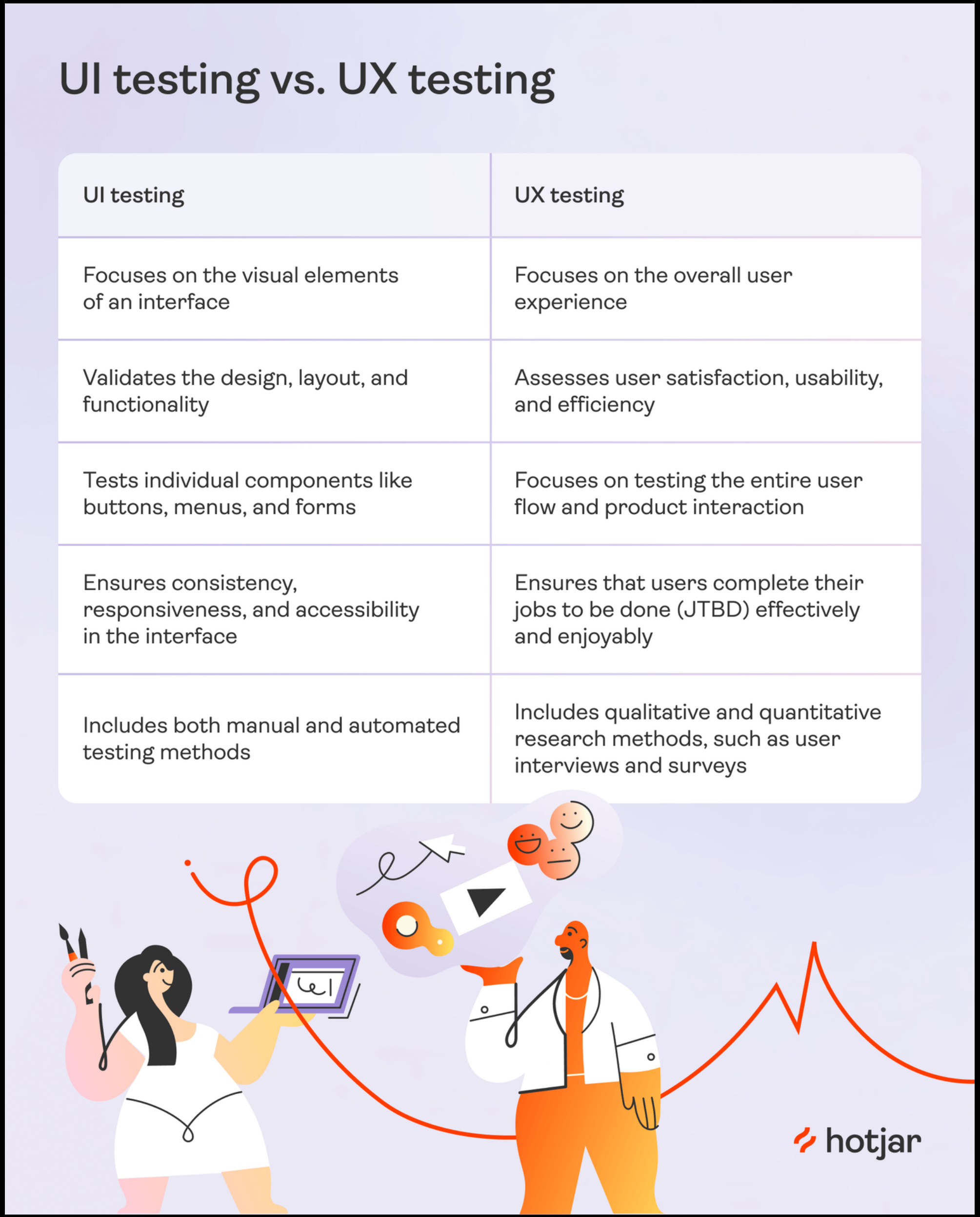
  Ensuring that user inputs are properly validated and that appropriate error messages or feedback are displayed to the user.

- **Performance and Responsiveness**

  Measuring the performance and responsiveness of the user interface, especially during high-load or complex interactions with the core.

Case Study Example

This case study demonstrates the implementation of user interface design principles and testing methodologies covered in the presentation. It highlights a successful project where a team effectively applied these concepts to create an intuitive and user-friendly application.



UI testing vs. UX testing

UI testing	UX testing
Focuses on the visual elements of an interface	Focuses on the overall user experience
Validates the design, layout, and functionality	Assesses user satisfaction, usability, and efficiency
Tests individual components like buttons, menus, and forms	Focuses on testing the entire user flow and product interaction
Ensures consistency, responsiveness, and accessibility in the interface	Ensures that users complete their jobs to be done (JTBD) effectively and enjoyably
Includes both manual and automated testing methods	Includes qualitative and quantitative research methods, such as user interviews and surveys

Conclusion

User Interface Design

Covered designing intuitive and responsive user interfaces for Quarkus applications using principles like separation of concerns, modularity, and accessibility.

Quarkus Core Integration

Discussed integrating the user interface with the core Quarkus application, leveraging the platform's features like dependency injection, reactive programming, and configuration management.

Comprehensive Testing

Implemented a robust testing strategy encompassing unit, integration, and end-to-end testing to ensure the overall quality and reliability of the Quarkus application.

The presentation provided a comprehensive overview of the key aspects involved in designing user interfaces, integrating with the Quarkus core, and implementing a comprehensive testing strategy, all while following the principles of hexagonal architecture.