Escuela de Ingenería Informática

Karl Deilmann

Programacíon de applicacíones moviles nativas

Report 4 - Choosing an Architecture

25 of Sep - 15 of Sep

ULPGC

Contents

1 Analysing Scenario **1**

1.1 Analysis 1

2 Peer-reviewing **7**

Chapter 1

Analysing Scenario

1.1 Analysis

Scenario 1: E-commerce Application for a Small Business

- **Budget**: Limited. The budget is constrained, implying cost-effective solutions are necessary.
- **Delivery Times**: 4 months. There is a relatively tight deadline, requiring a quick development cycle.
- **Human Resources**: One main developer and one designer. The team is small, which impacts the complexity of the project.
- **Performance Expectations**: Moderate traffic but a focus on speed and efficiency. The application should be responsive and efficient in handling user interactions.

Scenario 2: Interactive Social Application for a Startup

- **Budget**: Moderated. There is a reasonable budget available, allowing for more advanced features.
- **Delivery Times**: 6-8 months. The timeline is moderately flexible, allowing for a longer development cycle.
- **Human Resources**: A team of three developers, a designer, and a backend programmer. The team size is moderate, enabling the development of interactive features.

- **Performance Expectations**: High traffic and real-time interactions are expected. The application should handle these interactions seamlessly.

Scenario 3: Financial Application for a Large Enterprise

- **Budget**: High. There is a substantial budget allocated for the project, allowing for extensive development.
- **Delivery Times**: 10-12 months. The timeline is relatively flexible, accommodating a longer development cycle.
- **Human Resources**: A large team with multiple developers, designers, security specialists, and analysts. The project benefits from a well-staffed team with diverse expertise.
- **Performance Expectations**: Very high traffic, and security and efficiency are paramount. The application must be both secure and efficient.

Scenario 4: Health and Wellness Platform for Hospitals

- **Budget**: Very high. The project has a significant budget, enabling extensive development and security measures.
- **Delivery Times**: 12-15 months. The timeline is flexible, allowing for thorough development and testing.
- **Human Resources**: A multidisciplinary team comprising mobile developers, backend developers, security specialists, UX/UI designers, and systems analysts. The project benefits from a highly specialized team.
- **Performance Expectations**: Constant and high traffic due to a large number of patients. Data security and privacy are critical.

Scenario 5: Prototype Application for a Hackathon

- **Budget**: Minimal. The project has a minimal budget, emphasizing cost efficiency.
- **Delivery Times**: 48-72 hours. The timeline is extremely tight, requiring rapid development.

- **Human Resources**: A team of three students with mixed skills (developer, designer, and business). The team size is small and may have limited experience.
- **Performance Expectations**: As it's a prototype, no real traffic is expected. The focus is on creating a functional concept within a short time-frame.

Suggested Architectures

Scenario 1: E-commerce Application for a Small Business

- **Suggested Architecture:** MVC (Model-View-Controller).
- **Reasoning:**
  - **Investigate Architectures:** Before making a decision, it's crucial to understand the differences and advantages of common architectures like MVC. MVC provides a well-known and straightforward structure.
  - **Consider Requirements:** Not all applications require complex or robust architectures. In this case, with limited budget and time constraints, a simpler solution like MVC is more suitable.
  - **Prioritize Security:** While security is always important, in this scenario, where budget and time are limited, MVC doesn't hinder the implementation of basic security measures.
  - **Think About Scalability:** Although MVC is not as scalable as some other architectures, it can still handle moderate growth, which aligns with the expectations for this project.
  - **User Experience:** A responsive and efficient user interface is vital, and MVC allows for efficient development, contributing to a good user experience.
  - **Consult External Sources:** When in doubt, consulting external sources or real-world examples of MVC implementations can provide valuable insights and guidance.

Scenario 2: Interactive Social Application for a Startup

- **Suggested Architecture:** MVVM (Model-View-ViewModel).

- **Reasoning:**
  - **Investigate Architectures:** Understanding the strengths of MVVM is essential. It enables effective data binding, making it ideal for real-time interactive applications.
  - **Consider Requirements:** With a reasonable budget and flexibility in delivery times, MVVM's suitability for managing complex user interactions aligns well with the project's goals.
  - **Prioritize Security:** MVVM's data-binding capabilities can be leveraged to implement robust security features, crucial for an interactive social app.
  - **Think About Scalability:** MVVM's structured approach to managing UI-related logic and data makes it adaptable to the app's expected growth in user interactions.
  - **User Experience:** MVVM facilitates efficient UI updates, contributing to a positive user experience in a real-time application.
  - **Consult External Sources:** Exploring real-world examples of MVVM implementations in interactive social apps can provide valuable insights and best practices.

Scenario 3: Financial Application for a Large Enterprise

- **Suggested Architecture:** Clean Architecture.
- **Reasoning:**
  - **Investigate Architectures:** Clean Architecture offers modularity and maintainability, crucial for large-scale applications like financial platforms.
  - **Consider Requirements:** With a substantial budget and flexible delivery times, opting for a more robust architecture like Clean Architecture aligns with the project's complexity and long-term needs.
  - **Prioritize Security:** Clean Architecture's clear separation of concerns and controlled data flows make it an excellent choice for implementing stringent security measures in financial applications.
  - **Think About Scalability:** Clean Architecture's modularity makes it easy to scale and maintain, accommodating the expected high traffic and future growth.

  - **User Experience:** Clean Architecture ensures maintainable code, contributing to a reliable and error-free user experience, which is crucial in financial applications.
  - **Consult External Sources:** Exploring real-world financial applications built on Clean Architecture can provide valuable insights into best practices and compliance.

Scenario 4: Health and Wellness Platform for Hospitals

- **Suggested Architecture:** Hexagonal Architecture (Ports and Adapters).
- **Reasoning:**
  - **Investigate Architectures:** Hexagonal Architecture excels in maintaining data security and adaptability.
  - **Consider Requirements:** With a very high budget and flexible delivery times, choosing Hexagonal Architecture aligns with the project's complexity and the need to integrate with diverse hospital systems.
  - **Prioritize Security:** Hexagonal Architecture's emphasis on data security and clear boundaries makes it an excellent choice for safeguarding sensitive medical information.
  - **Think About Scalability:** Hexagonal Architecture's adaptability and ability to integrate with various systems support the platform's constant and high traffic expectations.
  - **User Experience:** Hexagonal Architecture's structured approach contributes to a reliable and user-friendly experience, critical in healthcare applications.
  - **Consult External Sources:** Examining real-world healthcare platforms built on Hexagonal Architecture can provide valuable insights into compliance and security measures.

Scenario 5: Prototype Application for a Hackathon

- **Suggested Architecture:** MVP (Model-View-Presenter).
- **Reasoning:**

- **Investigate Architectures:** In a hackathon setting, the focus is on rapid development, and MVP offers simplicity and speed.
- **Consider Requirements:** With minimal budget and a very tight timeline, MVP's straightforward structure aligns with the project's limited resources and need for quick results.
- **Prioritize Security:** While security is important, the primary goal in a hackathon is to create a functional prototype, and MVP allows the team to prioritize prototype development.
- **Think About Scalability:** Scalability is not a primary concern in a prototype scenario, and MVP provides a simple foundation for creating a proof of concept.
- **User Experience:** MVP focuses on the core functionality and can quickly demonstrate the application's concept, essential for hackathon presentations.
- **Consult External Sources:** During a hackathon, time is limited, and external sources may not be practical, but simple MVP examples can be found to guide development.

Chapter 2

Peer-reviewing