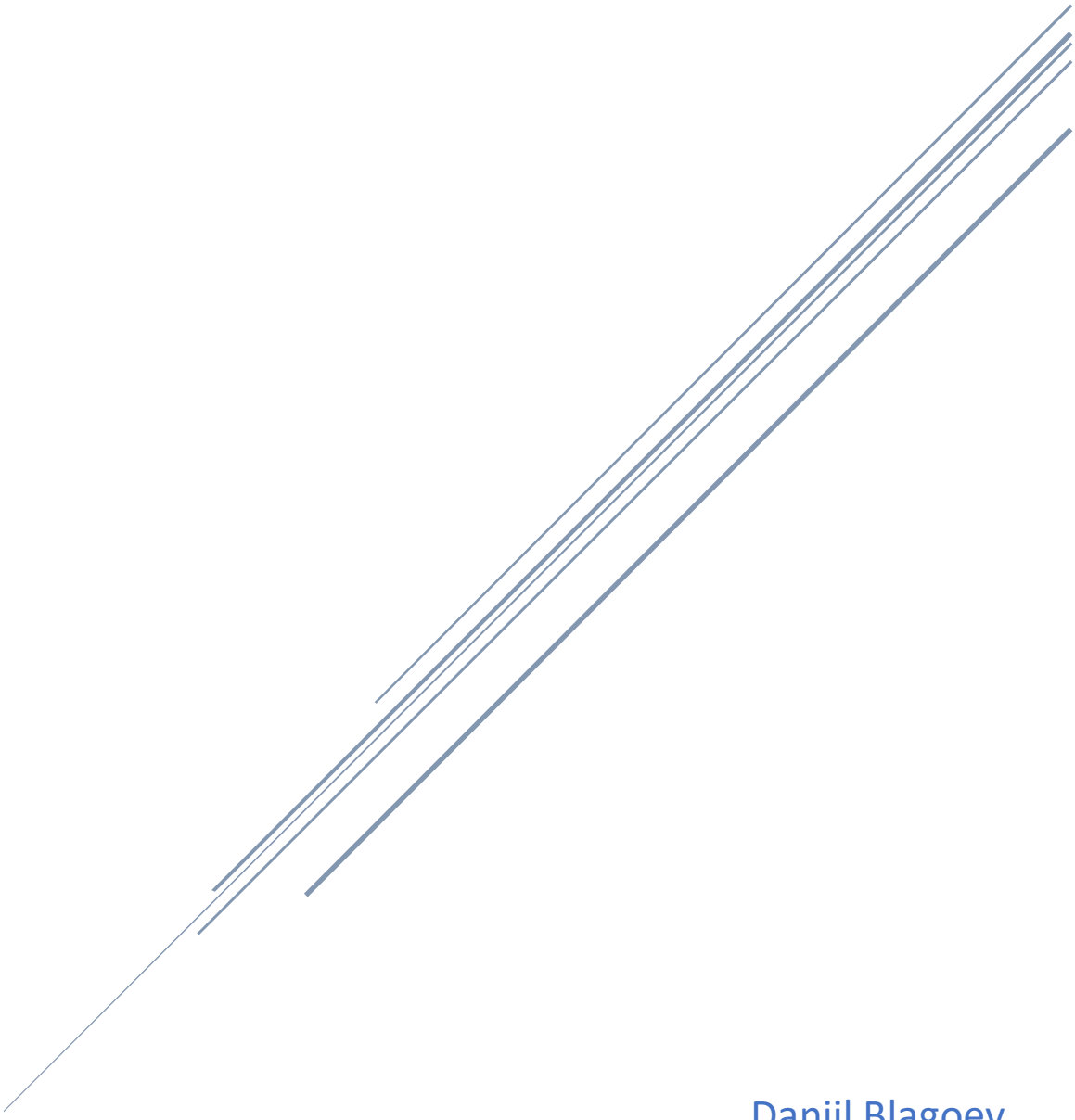
BITBY

Research document



Daniil Blagoev
Software Engineering- S3

Table of Contents

Research question-
What software architecture should I use for my Java project?

Sub-questions

- How many software architectures are there and what scenario is each architecture used in?
- Which software architectures are widely used in the Java community?
- What is the choice of software architecture for a project based on?
- What are the pros of the most used software architecture?
- What are the cons of the most used software architecture?
- Is my choice of software architecture good for my project?

Plan

- How many software architectures are there and what scenario is each architecture used in?
    - Literature study (Library)
    - Problem analysis (Field)
        - The literature study gives us a general overview of the types of architecture and through problem analysis the most fitting architecture can be chosen for each solution
- Which software architectures are widely used in the Java community?
    - Community research (Library)
    - Survey (Field)
        - Both methods collect information from people with experience in the field
- What is the choice of software architecture for a project based on?
    - SWOT analysis (Library)
    - Problem analysis (Field)
        - The positives and the negatives of each architecture need to be weighted in order to answer this question as accurately as possible
- What are the pros of the most used software architecture?
    - Community research (Library)
    - Domain modeling (Field)
        - Getting input from people who work with the architecture and creating a model to visualize the flow of the application will make it apparent where its strengths lie
- What are the pros of the most used software architecture?
    - Community research (Library)
    - Domain modeling (Field)
        - Getting input from people who work with the architecture and creating a model to visualize the flow of the application will make it apparent where its weaknesses lie
- Is my choice of software architecture good for my project?
    - Guideline conformity analysis (Showroom)
    - System test (Lab)
        - Making sure the system covers all requirements and is running with no issues would be one way to confirm the choice of architecture was correct

How did I get my answers?

- How many software architectures are there and what scenario is each architecture used in?
    - Went through multiple articles about software architecture and extracted the information summed up (e.g., Geeksforgeeks, Wikipedia)
- Which software architectures are widely used in the Java community?
    - Went through multiple articles and a survey (redhat, jrebel, oreilly)
- What is the choice of software architecture for a project based on?
    - Logically speaking, in order to solve a problem, it needs to be analyzed first, so in order to solve the problem of selecting an architecture, the requirements for the project have to be analyzed and the architecture which is most fitting is selected. If there is more than 1 architecture which does the job, preference can be taken into account.
- What are the pros of the most used software architecture?
    - Research in multiple articles (e.g., Wikipedia, redhat)
- What are the cons of the most used software architecture?
    - Research in multiple articles (e.g., Wikipedia, redhat)
- Is my choice of software architecture good for my project?
    - If the final product has no performance problems and it covers all requirements it is a good sign that the correct architecture was used.

Results

- How many architectures are there and what scenario is each architecture used in?
    - Layered architecture pattern
        - Applications that need to be built quickly
        - Enterprise applications that require traditional IT departments and processes
        - Applications that require strict standards of maintainability and testability
    - Event-driven pattern
        - For applications where individual data blocks interact with only a few modules
    - Microkernel pattern
        - Applications that have a clear segmentation between basic routines and higher-order rules
        - Applications that have a fixed set of core routines and dynamic set of rules that needs frequent updates
    - Microservices pattern
        - Businesses and web applications that require rapid development
        - Websites with small components, data centers with well-defined boundaries, and remote teams globally
    - Space-based pattern
        - Applications and software systems that function with a large user base and a constant load of requests
        - Applications that are supposed to address scalability and concurrency issues
    - Client-server pattern
        - Applications like emails, online banking services, the World Wide Web, network printing, file sharing applications, gaming apps, etc.

        - Applications that focus on real-time services like telecommunication apps are built with a distributed application structure
        - Applications that require controlled access and offer multiple services for a large number of distributed clients
        - An application with centralized resources and services that has to be distributed over multiple servers
    - Master-slave pattern
        - Development of Operating Systems that may require a multiprocessors compatible architecture
        - Advanced applications where larger services have to be decomposed into smaller components
        - Applications processing raw data stored in different servers over a distributed network
        - Web browsers that follow multithreading to increase its responsiveness
    - Pipe-filter pattern
        - It can be used for applications facilitating a simple, one-way data processing and transformation
        - Applications using tools like Electronic Data Interchange and External Dynamic List
        - Development of data compilers used for error-checking and syntax analysis
        - To perform advanced operations in Operating Systems like UNIX, where the output and input of programs are connected in a sequence
    - Broker pattern
        - Used in message broker software such as Apache ActiveMQ, Apache Kafka, RabbitMQ, and JBoss Messaging
        - For structuring distributed systems that have decoupled components
    - Peer-to-peer pattern
        - File-sharing networks such as Gnutella and G2
        - Cryptocurrency-based products such as Bitcoin and Blockchain
        - Multimedia products such as P2PTV and PDTP
    - Monolith pattern
        - The primary advantage is fast development speed due to the simplicity of having an application based on one code base
    - SOA pattern
        - Enterprise architects believe that SOA can help businesses respond more quickly and more cost-effectively to changing market conditions. This style of architecture promotes reuse at the macro (service) level rather than micro (classes) level.
- Which architectures are widely used in the Java community?
    - According to a survey conducted by jrebel.com, the most used architectures for 2022 are the following:
        - Microservices – 32%
        - Monolith – 22%
        - Modular monolith – 13%
        - SOA – 12%
        - Etc.
    - According to a few other articles found on Google (like Redhat's), the most used architecture is the Layered one
- What is the choice of architecture for a project based on?
    - Almost entirely based on what type of application is going to be built
    - It is based on personal preference in some cases

- What are the pros of the most used architecture?
    - Scaling up becomes easier
        - In the microservices architecture each service is designed, developed, and deployed independently. So, if one part of the software needs to be updated, we can update and use the microservice that handles that functionality
    - Leads to improved fault tolerance
        - Applications within microservices could continue working even if one service fails. This is because of the loose coupling between the services. Failure of one microservice does not affect the working of others
    - Ease of understanding of the codebase of the software system
        - It is easier to design a module keeping in mind the functionality of only that module. Understanding the specific functionality of each module is relatively more straightforward
    - Independent Deployment of each module
        - Since microservices are separate modules, they can be deployed independently in any application. If any module is modified, then the entire application need not be rebuilt and deployed
- What are the cons of the most used architecture?
    - Increased Complexity of Communication Between the Services
        - Splitting an application into multiple smaller modules increases the communication overhead. Developers must be extra cautious while handling requests between the different modules
    - Requires More Resources
        - With an increasing number of microservices, the resources needed to implement them increases as well
    - Global Testing and Debugging is Difficult
        - With microservices-based applications, each service needs to be launched and tested individually first. Then the application needs to be tested again, once all the services are launched
    - Relatively Complex Deployment
        - The deployment could be a complicated and challenging procedure. It would need coordination between multiple services during deployment. It would not be as simple as deploying a WAR file in a container
- Is my choice of software architecture good for my project?
    - Performance
        - The application has no issues with performance
    - Extensibility
        - The application is easily extensible due to the separation of concerns and separation by layer
    - Testability
        - The test coverage is close to 100% due to how accessible it is thanks to the separation by layer

Sources

Richards, M. (n.d.). *Software Architecture Patterns*. O'Reilly Online Learning. Retrieved October 16, 2022, from https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

Butani, A. (2022, October 11). *5 essential patterns of software architecture*. Enable Architect. Retrieved October 16, 2022, from https://www.redhat.com/architect/5-essential-patterns-software-architecture

GeeksforGeeks. (2021, October 27). *Types of Software Architecture Patterns*. Retrieved October 16, 2022, from https://www.geeksforgeeks.org/types-of-software-architecture-patterns/

2022 Java Architecture Trends. (2022, March 30). JRebel by Perforce. Retrieved October 11, 2022, from https://www.jrebel.com/blog/2022-java-architecture-trends

Dhaduk, H. (2022, June 6). *10 Best Software Architecture Patterns You Must Know About*. Simform - Product Engineering Company. Retrieved October 11, 2022, from https://www.simform.com/blog/software-architecture-patterns/

Atlassian. (n.d.). *Microservices vs. monolithic architecture*. Retrieved October 11, 2022, from https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

Wikipedia contributors. (2022, September 26). *Service-oriented architecture*. Wikipedia. Retrieved October 11, 2022, from https://en.wikipedia.org/wiki/Service-oriented_architecture

What are Microservices and Their Advantages and Disadvantages? (2022, October 5). HitechNectar. Retrieved October 11, 2022, from https://www.hitechnectar.com/blogs/5-pros-and-cons-of-microservices-explained/

Conclusion

In my personal opinion, the layered would be a correct choice of architecture for my crypto exchange project due to several reasons: higher ease of development, better testability, and due to me being more familiar with it compared to the rest of the architectures.