MINISTERUL EDUCAŢIEI ŞI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Admis la susținere

Şef de departament:

Fiodorov I. dr., conf.univ.

_____

„___”_____2022

Analiza arhitecturilor monolitice și microservicii pentru proiectarea aplicațiilor web

Teză de master

Student: **Alexandru Jucov, TI-201M**

Coordonator: **Antohi Ionel, asistent univ. mag.**

Consultant: **Cojocaru Svetlana, lector univ. mag.**

Chișinău, 2022

АННОТАЦИЯ

Наличие правильной архитектуры для системы является решающим фактором для определения будущей ремонтопригодности, улучшений и масштабируемости.

В первой главе проанализированы менее популярные архитектурные решения, хотя они до сих пор используются в некоторых приложениях выявлены их достоинства и недостатки. Цель первой главы - проанализировать типы архитектур монолитного программного обеспечения и микросервисов и сделать вывод о том, какая из них более подходит для разработки веб-приложений для малых и средних предприятий. Анализ учитывает сильные и слабые стороны каждой архитектуры. В конце диссертации было принято объективное решение относительно того, что ныне модная микросервисная архитектура не является панацеей от всех проблем и множество решений которая она предлагает работают в основном на больших проектах. Маленьким и средним проектам цель которых выйти на рынок как более скорей не следует использовать микросервисную архитектуру, а лучше сосредоточиться на монолитной архитектуре, которая позволят быстро вносить радикальные изменения и проще в тестируемости.

Во второй главе продемонстрированы шаблоны проектирования, подходы и инструменты для реализации микросервисной архитектуры в серверных приложениях. В этой главе рассмотрены стратегии декомпозиции, а именно как построить микросервисную архитектуру и как стоит разбивать систему на микросервисы. Также рассмотрены межпроцессорные взаимодействия, проанализированы синхронные и асинхронные методы общения между микросервисами, рассмотрены механизмы коммуникации на основе запросов/ответов такие как REST и gRPC выявлены их достоинства и недостатки. Рассмотрены шаблоны проектирования и подходы, которые решают проблемы с обнаружением микросервисов и восстановление после отказа. Так же в этой главе проанализированы инструменты для асинхронного общения на основе стандартов AMQP и STOMP такие как ActiveMQ, RabbitMQ, Apache Kafka, рассмотрены их достоинства и недостатки. Так же проанализированы решения проблем, связанные с дублированием сообщений и устареванием сообщений.

В третей главе рассмотрены библиотеки и фреймворки для создания серверных приложений на основе микросервисной архитектуры для языка программирования C#. Продемонстрирован реализация минимально жизнеспособного продукта электронное правительство, который будет в себя включать онлайн голосование, онлайн идентификатор гражданина, права на вождение механического средства, технический паспорт механического средства. Так же рассмотрены проблемы при реализации данного проекта.

REZUMAT

A avea arhitectura potrivită pentru un sistem este esențial pentru determinarea întreținerii, îmbunătățirilor și scalabilității viitoare.

În primul capitol, sunt analizate soluțiile arhitecturale mai puțin populare, deși sunt încă utilizate în unele aplicații, avantajele și dezavantajele lor sunt dezvăluite. Scopul primului capitol este de a analiza tipurile de software monolitic și arhitecturi de microservicii și de a concluziona care dintre acestea este mai potrivit pentru dezvoltarea aplicațiilor web pentru întreprinderile mici și mijlocii. Analiza ia în considerare punctele tari și punctele slabe ale fiecărei arhitecturi. La sfârșitul disertației, a fost luată o decizie obiectivă cu privire la faptul că acum arhitectura la modă a microserviciului nu este un panaceu pentru toate problemele și numeroasele soluții pe care le oferă funcționează în principal pe proiecte mari. Proiectele mici și mijlocii al căror obiectiv este de a intra pe piață cât mai curând posibil nu ar trebui să utilizeze o arhitectură de microserviciu, ci mai degrabă să se concentreze pe o arhitectură monolitică care să permită schimbări radicale rapide și o testabilitate mai ușoară.

Al doilea capitol demonstrează modele de proiectare, abordări și instrumente pentru implementarea arhitecturii microservice în aplicații server. Acest capitol discută strategiile de descompunere, și anume cum să construim o arhitectură de microservice și cum să împărțim un sistem în microservicii. De asemenea, sunt luate în considerare interacțiunile interprocesorului, sunt analizate metodele de comunicare sincrone și asincrone între microservicii, sunt luate în considerare mecanismele de comunicare bazate pe cereri / răspunsuri precum REST și gRPC, sunt identificate avantajele și dezavantajele acestora. Discută modele de proiectare și abordări care abordează problemele de descoperire și de reluare a microserviciilor. De asemenea, în acest capitol, sunt analizate instrumentele pentru comunicarea asincronă bazate pe standardele AMQP și STOMP, precum ActiveMQ, RabbitMQ, Apache Kafka, sunt luate în considerare avantajele și dezavantajele acestora. Sunt analizate, de asemenea, soluțiile la problemele asociate cu mesajele duplicate și caducitatea mesajelor.

Al treilea capitol discută bibliotecile și cadrele pentru crearea de aplicații server bazate pe o arhitectură de microserviciu pentru limbajul de programare C #. A demonstrat implementarea produsului minim viabil de e-guvernare, care va include votul online, ID-ul cetățeanului online, permisul de conducere pentru un vehicul mecanic, pașaportul tehnic al unui vehicul mecanic. Sunt luate în considerare și probleme în implementarea acestui proiect.

ABSTRACT

Having the right architecture for a system is critical to determining future maintainability, enhancements, and scalability.

In the first chapter, less popular architectural solutions are analyzed, although they are still used in some applications, their advantages and disadvantages are revealed. The purpose of the first chapter is to analyze the types of monolithic software and microservices architectures and conclude which one is more suitable for developing web applications for small and medium enterprises. The analysis takes into account the strengths and weaknesses of each architecture. At the end of the dissertation, an objective decision was made regarding the fact that the now fashionable microservice architecture is not a panacea for all problems and the many solutions that it offers work mainly on large projects. Small and medium-sized projects whose goal is to go to market as soon as possible should not use a microservice architecture, but rather focus on a monolithic architecture that will allow for quick radical changes and easier testability.

The second chapter demonstrates design patterns, approaches, and tools for implementing microservice architecture in server applications. This chapter discusses decomposition strategies, namely how to build a microservice architecture and how to break a system into microservices. Also, interprocessor interactions are considered, synchronous and asynchronous methods of communication between microservices are analyzed, communication mechanisms based on requests / responses such as REST and gRPC are considered, their advantages and disadvantages are identified. It discusses design patterns and approaches that address microservice discovery and failover issues. Also in this chapter, tools for asynchronous communication based on AMQP and STOMP standards, such as ActiveMQ, RabbitMQ, Apache Kafka, are analyzed, their advantages and disadvantages are considered. The solutions to the problems associated with duplicate messages and obsolescence of messages are also analyzed.

The third chapter discusses libraries and frameworks for creating server applications based on a microservice architecture for the C # programming language. Demonstrated the implementation of the minimum viable e-government product, which will include online voting, online citizen ID, driving license for a mechanical vehicle, technical passport of a mechanical vehicle. Problems in the implementation of this project are also considered.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В течение долгого времени индустрия программного обеспечения придерживалась мнения, что архитектура - это то, что необходимо разработать и завершить до написания первой строчки кода. Вдохновленный строительной отраслью, считалось, что признаком успешной архитектуры программного обеспечения было то, что не нужно было менять во время разработки, часто это реакция на высокую стоимость брака и переделок, которые могут возникнуть из-за события изменения архитектуры. Это видение архитектуры было резко оспорено появлением гибких методов разработки программного обеспечения. Подход к заранее спланированной архитектуре был основан на понятии, что требования также должны быть исправлены до начала кодирования, что привело к поэтапному (или водопадному) подходу, когда за требованиями следовала архитектура, за которой следовало построение (программирование). Однако гибкий мир поставил под сомнение само понятие фиксированных требований, заметив, что регулярные изменения в требованиях являются бизнес-необходимостью в современном мире, и предоставил методы планирования проектов, учитывающие контролируемые изменения. В этом новом гибком мире многие люди сомневались в роли архитектуры. И, конечно же, заранее спланированное архитектурное видение не могло соответствовать современному динамизму. Но есть и другой подход к архитектуре, который предполагает гибкие изменения. В этом представлении архитектура - это постоянное усилие, которое тесно связано с программированием, так что архитектура может реагировать как на изменяющиеся требования, так и на обратную связь от программирования.

Если вы разрабатываете и запускаете онлайн-продукт, вы рано или поздно столкнетесь с идеей архитектуры веб-приложений . Разработчики программного обеспечения используют это слово для обозначения высокоуровневой структуры цифрового продукта, которая включает в себя хранилище данных и детали работы сервера.

Надежность, производительность и безопасность вашего продукта определяются качеством веб-инфраструктуры. В эпоху стремительных процессов никто не знает, что будет с их продуктом в будущем. Однажды популярность приложения может вырасти, резко увеличивая количество пользователей. Люди также могут начать более активно использовать некоторые периферийные функции или попросить добавить больше функций в отзывах. На этом этапе архитектура определяет способность приложения расти вместе с вашим бизнесом и решать новые проблемы по мере его изменения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Kevin Hoffman. Building Microservices with ASP.NET Core. Sebastopol: O'Reilly. ISBN 063-692-005-207-4.
2. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sornmerlad. Michael Stal PATTERN-ORIENTED SOFTWARE ARCHITECTURE. New York: John Wiley & Sons. ISBN 0-471-95889-7.
3. Neal Ford, Rebecca Parsons, Patrik Kua. Building Evolutionary Architectures. Sebastopol: O'Reilly. ISBN 978-149-198-636-3.
4. Chris Richardson. Microservices Patterns. Shelter Island: Manning Publications. ISBN 978-161-729-454-9.