Revolutionizing the Job Search: A Scalable Online Job Portal Powered by Distributed Systems and Cloud Computing

Tchamabe Martine Carole
ICTU202223095

KENGNE MBOU BRICE JUNIOR
ICTU20222938

FON CHRIS BRIGHT
ICTU20222969

Aghuke De Ngeh Briyand
ICTU20222984

NDE HURICH DILAN
ICTU20223351

MBAH ATANGANA DAVID
ICTU20223510

CHI TITI ROSITA LOUlIA
ICTU20223194

Babila Hans Nteh
ICTU20233978

Kaldapa Zokom Akim Frederic
ICTU20223332

Eko Pearleen
ICTU20223530

Abingseh Cindy

ICTU20223342

11. August 2024

Supervisor:
Dr.-Ing. Philippe Tamla
Visiting Professor at ICT University

PostDoc Researcher at
Department of Multimedia and Internet Applications
Faculty of Mathematics und Computer Science
Universitätsstr. 1
58097 Hagen

Abstract

This section should be written last, as it summarizes all the results of the work in a compressed form. The length should not exceed 100 words or half a page. The table of contents should also be limited to about half a page so that it fits on the back of the cover page along with the abstract. If necessary, the depth of the outline should be adjusted accordingly when generating the table of contents.

...

Inhaltsverzeichnis

List of Figures

List of Tables

List of Source Code

1 Introduction

Whether entering the job market for the first time or re-entering after a break or switching career, job search is a challenging task [BNK20]. However, imagine having applications that transform this challenging process into a more accessible, organized, and user-friendly experience for both employers and candidates. **Faä** aims to revolutionize the recruitment process by leveraging the principles of **Cloud Computing and Distributed Systems**. This project is designed to provide a comprehensive platform where job seekers and employers can engage in seamless recruitment activities. By utilizing **Distributed Systems**. The Nunamaker's [AC04] Multi-methodological Approach methodology implemented is an effective model for the sole purpose of a web job portal creation.

1.1 Motivation

We will now motivate our work by introducing relevant research and development associated with the challenges of developing a **Job Portal** system using **Distributed Systems and Cloud Computing**. **Microservices Architecture for Cloud Computing** [Pac16] presents a proposal for developing **microservices-oriented** software that can be delivered through the **cloud**.

1.2 Problem Statement

Novice users may face challenges in effectively utilizing Job Portal system, particularly due to the technical and conceptual knowledge required for navigating complex features like job matching algorithms, user data management. Additionally, current Job Portal may not be able to seamlessly scale to accommodate large-scale user traffic and data volumes, while ensuring high availability, low latency, and robust data management capabilities.

1.3 Research Questions

This leads to the following Research Question (RQs):

- **RQ1**: **How can the microservices-based architecture of a job portal be optimized to deliver fast and efficient responses to user requests, ensuring a responsive user experience?**

1.4 Research Method

To address this RQ, we will employ the well-established **Nunamaker methodology**[AC04], a systematic approach to developing information systems. This methodology involves delineating 04 Research Objectives (ROs) for the RQ, covering the stages of observation, theory building, implementation, and evaluation.

1.5 Research Objectives

The primary objective of this research is to design, implement, and evaluate a highly **scalable**, fault-tolerant, and cost-effective **distributed system architecture** for a job portal website, leveraging **cloud computing** and other advanced distributed systems technologies.

1.6 Approach

Using the Nunamaker framework, our Job Portal project is structured across four key phases: **Observation**, **Theory Building**, **Systems Development**, and **Experimentation**. In the **Observation phase**, which will be detailed in Chapter 2, we will conduct a comprehensive review of existing recruitment processes and technologies to identify challenges (RO1). The **Theory Building** phase, will be covered in Chapter 3, it will involves developing models to optimize the system's architecture for performance and scalability (RO2). In **Systems Development**, that will be outlined in Chapter 4, we will implement the Job Portal based on these models, focusing on cloud deployment and user-centric features (RO3). Finally, the **Experimentation phase**, will be described in Chapter 5, evaluates the system's performance and usability through testing and user feedback (RO4). This structured approach ensures a systematic and effective development process.

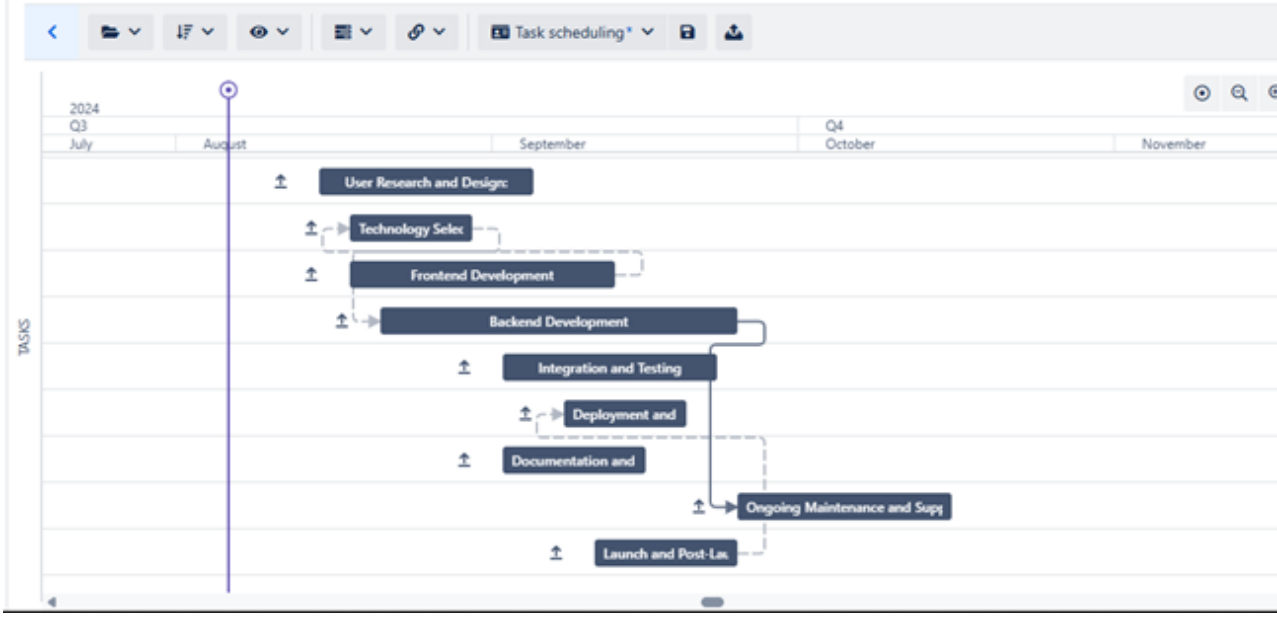
1.7 Outline

Our research project uses the **Nunamaker methodology** to build a scalable, user-friendly Job Portal system using **cloud computing** and **distributed systems**. The project's goals and research topics are outlined in Chapter 1 of the introduction. The Observation phase is covered in Chapter 2, whereby RO1 is addressed through an analysis of current **recruitment technologies**. In Chapter 3, Theory Building is covered, and RO2 entails creating models to **optimize the design of the system**. In the Systems Development phase, Chapter 4 discusses RO3 and provides design

and implementation details for the system. Lastly, Chapter 5 discusses the **Experimentation stage**, assessing the functionality and performance of the system and covering RO4.

1.8 Work and Schedule

Here is our Gantt chart of the work packages and their temporal placement within the overall timeframe.



2 State of the Art in Science and Technology

2.1 Relevant Technologies

In the rapidly evolving landscape of web-based job portals, the need for scalable, efficient, and integrated platforms has become increasingly paramount. As organizations seek to harness the power of emerging technologies to streamline the job search and hiring processes, the role of distributed systems architecture has taken center stage. Within this context, the synergistic integration of Python for the backend and React for the frontend has emerged as a compelling approach to building robust and adaptable job portal applications.

Python's Prowess in Distributed Systems and Cloud Computing

Python, the versatile and widely-adopted programming language, has become a go-to choice for building the backend components of distributed systems, particularly in the realm of job portal applications [Gri18]. Its inherent scalability and concurrency features, facilitated by frameworks like Django, enable job portal systems to handle high-volume traffic and concurrent client connections [HK09]. This scalability is crucial for accommodating the growing user base and data demands of modern job portals.

Furthermore, Python's seamless integration with cloud computing services, such as AWS SDK for Python and Google Cloud SDK, allows job portal applications to leverage the benefits of cloud infrastructure [Ser23]. This cloud-native approach empowers job portals to scale their resources elastically, store and process vast amounts of data, and leverage serverless computing capabilities, all of which are essential for delivering a robust and reliable user experience.

Equally important is Python's prowess in data processing and analytics, which is fundamental to the job portal domain. The language's extensive library ecosystem, including powerful data manipulation and machine learning tools, enables job portal backends to effectively process, index, and analyze user profiles, job postings, and application data [McK17]. This, in turn, fuels features like personalized job recommendations, advanced search functionalities, and comprehensive reporting capabilities.

Strengths and Limitations of Python in Job Portal Applications

The suitability of Python for job portal applications is underpinned by its strengths in scalability, cloud integration, and data-centric capabilities [RD09]. However, it is essential to acknowledge the potential limitations as well. While Python is generally fast enough for most web applications, it may not be the optimal choice for highly performance-sensitive tasks, such as real-time job search or complex job matching algorithms [Lut13]. Additionally, the initial learning curve for non-Python developers may present a challenge, and the management of Python's vast ecosystem of libraries and packages can become complex, especially in distributed systems with multiple components [Sla19].

React's Role in Distributed Systems and Cloud Computing for Job Portals

Complementing the backend capabilities of Python, React, the popular JavaScript library for building user interfaces, plays a crucial role in the distributed system architecture of job portal applications. React's modular and component-based design aligns seamlessly with the distributed nature of job portal systems, enabling the creation of reusable, scalable, and maintainable user interface components [Tab17].

React's efficient DOM manipulation, facilitated by its virtual DOM and diffing algorithms, ensures smooth and responsive user experiences, a vital aspect of job portal applications that require seamless interactions and real-time updates [Bod19]. Additionally, React's flexibility in consuming and rendering data from backend APIs, such as those provided by the Python-based job portal backend, facilitates the creation of a cohesive and integrated distributed system [Cor19].

The scalability and performance-centric nature of React make it well-suited for building job portal applications that need to accommodate growing user bases and data volumes [Faw19]. Furthermore, the extensive React ecosystem, including a wide range of supporting libraries, tools, and development environments, enables the job portal development team to leverage best practices, accelerate development, and ensure code maintainability [Kho18].

Strengths and Limitations of React in Job Portal Applications

The strengths of React in job portal applications are manifold, including its modular and component-based design, efficient DOM manipulation, seamless integration with backend APIs, scalability, and the rich ecosystem of supporting tools and libraries [Zak18]. However, it is essential to acknowledge that the initial learning curve for non-React developers may present a challenge, and the complexity of managing application state in large-scale job portal applications can require additional architectural patterns and tooling [AB17]. Additionally, while React is generally performant, highly complex or data-intensive user interfaces in the job portal application may still require careful optimization and performance monitoring [Ric19].

Conclusion

By seamlessly integrating the backend capabilities of Python and the frontend prowess of React, job portal web applications can be built as scalable, efficient, and integrated distributed systems that harness the power of cloud computing and cater

to the specific requirements of the job portal domain [Gac15]. This synergistic approach allows job portal developers to leverage the strengths of both technologies, mitigate their respective limitations, and deliver a comprehensive and user-centric platform that drives innovation in the job search and hiring landscape.

3 Modeling

4 Implementation

5 Evaluation

6 Summary and Outlook

Literatur

[AB17] Alex Accomazzo und Adam Bucks. *React and React Native*. Packt Publishing, 2017.

[AC04] Lascelles Adams und James Courtney. »Achieving Relevance in IS Research via the DAGS Framework.« In: Bd. 37. Jan. 2004. DOI: 10.1109/HICSS.2004.1265615.

[BNK20] Saša Baškarada, Vivian Nguyen und Andy Koronios. »Architecting microservices: Practical opportunities and challenges«. In: *Journal of Computer Information Systems* (2020).

[Bod19] Azat Boduch. *React Quickly: Painless Web Apps with React, JSX, Redux, and GraphQL*. Manning Publications, 2019.

[Cor19] Shama Hoque Cory. *Full-Stack React Projects: Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node.js*. Packt Publishing, 2019.

[Faw19] Mark Fawcett. *React in Action*. Manning Publications, 2019.

[Gac15] Cory Gackenheimer. *React Components*. Apress, 2015.

[Gri18] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc., 2018.

[HK09] Adrian Holovaty und Jacob Kaplan-Moss. *The Django Book*. Djangobook.com, 2009.

[Kho18] Alex Khor. *React Cookbook: Proven Recipes for Building Robust Apps with React 16*. O'Reilly Media, Inc., 2018.

[Lut13] Mark Lutz. *Learning Python*. O'Reilly Media, Inc., 2013.

[McK17] Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2017.

[Pac16] Vinod Keshaorao Pachghare. »Microservices architecture for cloud computing«. In: *architecture* 3 (2016), S. 4.

[RD09] Guido van Rossum und Fred L. Drake. *The Python Language Reference*. Python Software Foundation, 2009.

[Ric19] Adam Richter. *Pro React 16*. Apress, 2019.

[Ser23] Amazon Web Services. *Boto3 Documentation*. 2023. URL: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html.

[Sla19] Brett Slatkin. *Effective Python: 90 Specific Ways to Write Better Python*. Addison-Wesley Professional, 2019.

[Tab17] Noah Taber. *Building Modular Applications with Mvc*. Packt Publishing, 2017.

[Zak18] Nicholas C. Zakas. *Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers*. No Starch Press, 2018.

Glossary

LMS Online learning management systems *siehe* Online learning management systems

Online learning management systems provided digital platforms for managing courses, materials, and student interactions 9