Hochschule Fulda
University of Applied Sciences

Distributed Applications - AI5109

Implementation and Deployment

(Module 3)

21.jan.2024

Table of Contents

Abstract

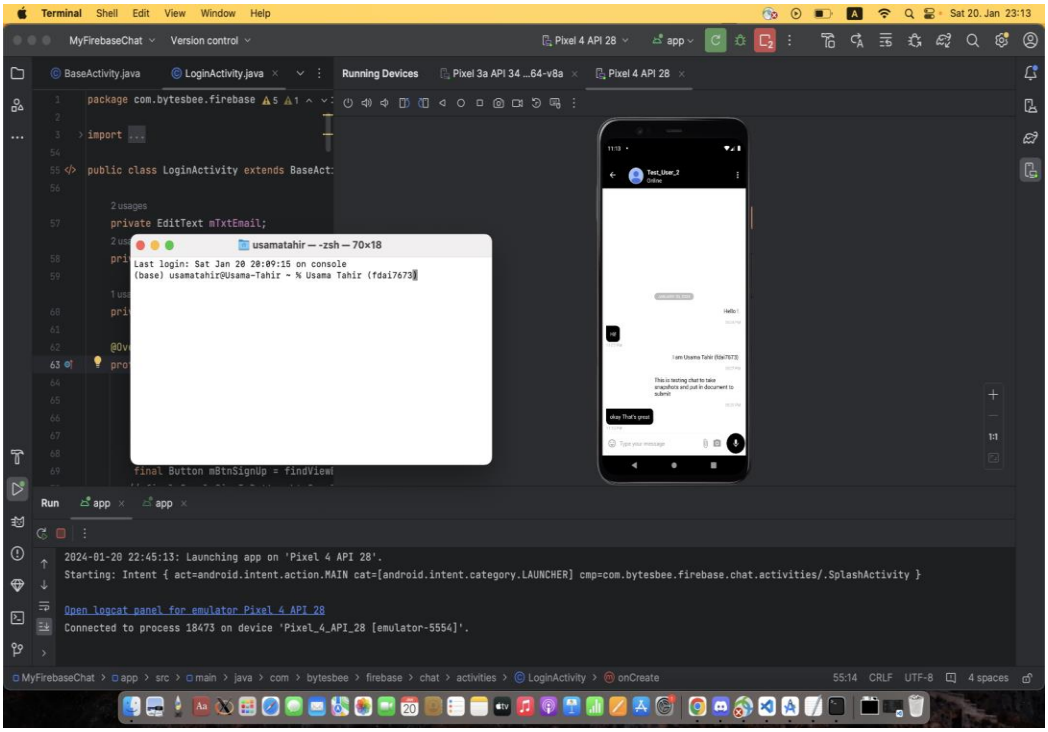
The "What's Chat" Android application is a modern and feature-rich chat platform developed using Android Studio. This cutting-edge chat application leverages the robust capabilities of Google Firebase as its backend infrastructure, tapping into key features such as authentication and Firestore for seamless real-time data management.

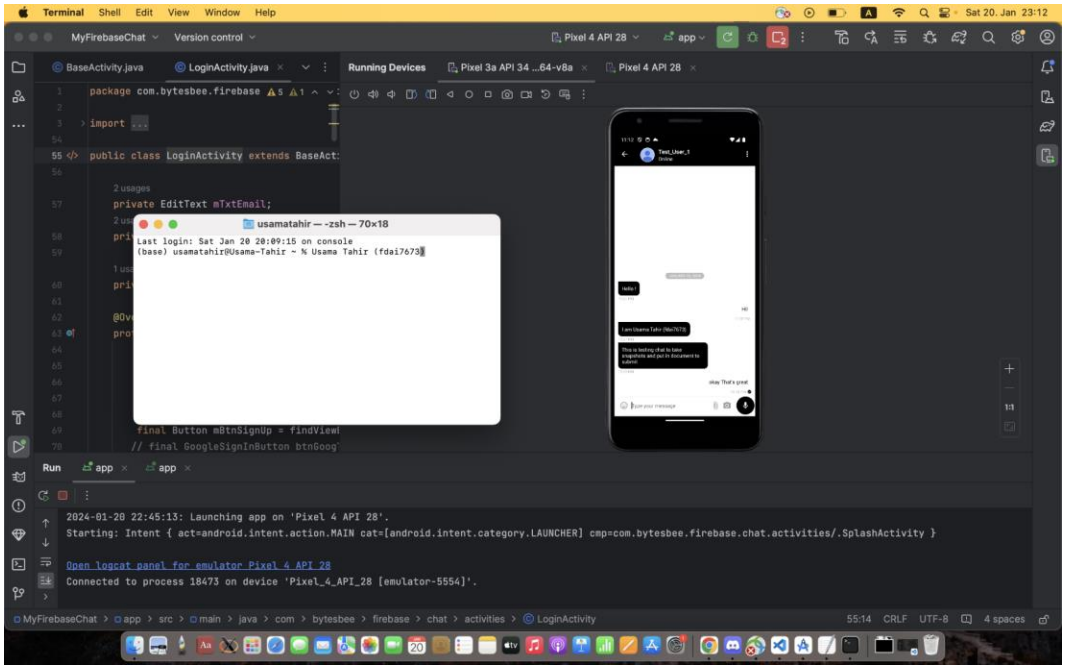
Key Features:

- **Authentication:** What's Chat ensures secure user registration and login through the implementation of Firebase Authentication. Users can confidently create accounts and access the app with ease, offering a foundation for a secure and personalized chat experience.
- **Firestore Integration:** The application harnesses the power of Firebase Firestore to facilitate real-time messaging. Firestore's NoSQL database allows for efficient and scalable storage, retrieval, and synchronization of chat data, ensuring a smooth and responsive user experience.
- **Push Notifications:** Integration with Firebase Cloud Messaging (FCM) ensures that users stay connected even when the app is not actively in use. Push notifications keep users informed about new messages, ensuring timely responses and enhancing overall engagement.
- **Media Sharing:** What's Chat goes beyond simple text messaging by allowing users to share images and other media files. This feature adds a multimedia dimension to conversations, enriching the user experience and facilitating more dynamic communication.

The "What's Chat" Android application stands as a testament to the seamless integration of cutting-edge technologies, providing a user-friendly, secure, and feature-complete chat experience. By combining the power of Android Studio and Google Firebase, the application sets a new standard for modern chat applications, offering a platform where users can connect, communicate, and share in a dynamic and engaging environment.

Snapshots of Application





Answers to the questions:

What is an application server? what are its uses

An application server is a type of software framework that provides a runtime environment for the execution of applications. It is specifically designed to host and run applications, handling various tasks related to application deployment, management, and performance optimization. Application servers play a crucial role in the architecture of many web and enterprise applications. Here are some key aspects and uses of application servers

Execution Environment: Application servers provide a runtime environment where applications can run. They manage resources, handle communication between components, and ensure that the application code can execute smoothly.

Middleware Services: Application servers often include middleware services that facilitate communication and interaction between different software components. This can involve services like message queuing, transaction processing, and database connectivity.

Scalability: Application servers help in scaling applications by distributing the load across multiple instances or nodes. This ensures that the application can handle increased traffic and demands, providing a scalable solution for growing user bases.

Connection Handling: They manage the connections between clients and applications. This includes handling multiple concurrent connections, managing session state, and ensuring data integrity during communication.

Security: Application servers often incorporate security features, such as authentication and authorization mechanisms, to protect applications and data from unauthorized access. They can also provide encryption and secure communication channels.

Transaction Management: For applications that involve transactions (such as financial transactions), application servers offer transaction management services to ensure that operations are completed successfully or rolled back in case of failures.**Load Balancing:** In a distributed environment, application servers may implement load balancing to distribute incoming requests evenly across multiple servers. This helps in optimizing resource utilization and improving overall system performance.

Caching and Optimization: Application servers may include caching mechanisms to store frequently accessed data, reducing the need to retrieve it from the database repeatedly. This improves response times and overall system performance.

Integration with Databases: Application servers often integrate with databases, handling the connection pool, executing queries, and managing transactions between the application and the database.

Support for Different Protocols: Application servers support various communication protocols, such as HTTP, HTTPS, and others, making them suitable for web applications and other distributed systems.

Briefly explain Serverless Computing

Serverless computing, also known as Function as a Service (FaaS), is a cloud computing execution model where cloud providers automatically manage the infrastructure required to run applications. In a serverless architecture, developers focus on writing code in the form of functions, and the cloud provider takes care of provisioning, scaling, and managing the servers that execute those functions. The term "serverless" doesn't mean there are no servers involved, but rather that developers don't need to explicitly manage or provision them.

Key characteristics of serverless computing include:

Event-Driven: Serverless applications are often event-driven, responding to triggers or events such as HTTP requests, database changes, file uploads, etc. Each function is designed to execute in response to a specific event.

Automatic Scaling: Serverless platforms automatically scale the number of function instances based on demand. If there's a sudden increase in the number of requests, the platform can quickly scale up to handle the load, and scale down during periods of inactivity.

Pay-Per-Use Pricing: With serverless computing, users are billed based on the actual resources consumed by their functions. Since the infrastructure is automatically managed, users don't need to pay for idle time or reserved capacity.

Stateless Functions: Serverless functions are designed to be stateless, meaning they don't store information between executions. Any required state or data is typically stored externally, such as in a database or object storage.

Short-Lived Execution: Functions in serverless computing are designed to execute for a brief duration to perform a specific task. Long-running or continuous processes may not be well-suited for the serverless model.

Abstraction of Infrastructure: Developers work at a higher level of abstraction, focusing on writing code and defining function triggers, while the cloud provider manages the underlying infrastructure, such as servers, networking, and operating systems.

Explain Distributed Mobile Computing

Distributed Mobile Computing refers to the concept of distributing computing tasks and processing across multiple mobile devices that are interconnected. This approach leverages the combined computational power and resources of a network of mobile devices to perform complex tasks, share information, and collaboratively solve problems. The goal is to enhance the overall performance and capabilities of mobile applications by harnessing the collective power of a distributed system.

Key characteristics and concepts of Distributed Mobile Computing include:

Collaborative Processing: Mobile devices in a distributed system collaborate to share computational tasks. This collaboration can involve dividing a large computation into smaller tasks and distributing them across multiple devices.

Resource Sharing: Each mobile device in the network contributes its resources, such as processing power, memory, and storage, to support the overall computing needs of the system. This resource pooling enables more efficient use of available capabilities.

Data Sharing and Synchronization: Distributed mobile computing often involves sharing and synchronizing data across devices. This can include distributing data for processing, aggregating results, and maintaining consistency among devices.

Decentralized Control: Unlike centralized computing models where a single server or system manages the entire process, distributed mobile computing typically involves decentralized control. Each device may have some level of autonomy and makes decisions based on local information.

Ad Hoc Networking: Devices in a distributed mobile computing system may form ad hoc networks, allowing them to communicate directly with each other without relying on a

centralized infrastructure. This is particularly useful in scenarios where network connectivity is dynamic or unreliable.

Load Balancing: Load balancing is crucial in distributed systems to ensure that computational tasks are evenly distributed among devices. This helps in optimizing the overall system performance and resource utilization.

Fault Tolerance: Distributed mobile computing systems often incorporate mechanisms for fault tolerance. If a mobile device fails or becomes unavailable, the system can adapt by redistributing tasks to other devices to maintain continuous operation.

Mobile Cloud Computing: Distributed mobile computing is often associated with mobile cloud computing, where mobile devices leverage cloud resources for additional storage, computation, and services. This extends the capabilities of mobile devices beyond their intrinsic limits.