TWMailerPro Protokoll

Client- und Server-Architektur

Der TWMailerPro besteht aus einer Client-Server-Architektur auf, wobei der Server als zentrale Schnittstelle für die Verarbeitung von Anfragen dient, die von Clients über TCP/IP-Verbindungen gesendet werden. Der Server verwaltet die Authentifizierung, das Speichern, Auflisten, Lesen und Löschen von E-Mail-Nachrichten. Clients bieten eine Benutzerschnittstelle, über die Nutzer mit dem Server interagieren, um E-Mails zu senden, abzurufen und zu verwalten.

Verwendete Technologien und Bibliotheken

Die Entwicklung des Systems erfolgte in C++ unter Verwendung von Standardbibliotheken für die Netzwerkkommunikation (sys/socket.h, netinet/in.h, arpa/inet.h, unistd.h), sowie für Ein- und Ausgabeoperationen (iostream, fstream). Zusätzlich wurden die Bibliotheken string und vector für die Verarbeitung von Zeichenketten und dynamischen Datenstrukturen genutzt. Die Header-Datei sys/wait.h wurde für die Prozesskontrolle und -synchronisation verwendet.

Entwicklungsstrategie

Der TWMailerPro basiert auf klar definierten Befehlen (LOGIN, SEND, LIST, READ, DEL) für eine einfache Interaktion zwischen Client und Server. Es ermöglicht eine strukturierte Kommunikation und erleichtert die Erweiterung des Systems. Zudem integriert die Sicherheitsfunktionen LDAP-Authentifizierung und Blacklist-Mechanismen für zusätzlichen Schutz. Diese Kombination sorgt für eine klare und sichere Umgebung, indem sie sicherstellt, dass nur verifizierte Benutzer Zugang erhalten und unautorisierte Zugriffsversuche nach dreimaligem Fehlschlagen durch eine 60-sekündige Sperre unterbunden werden.

Verwendete Synchronisationsmethoden

Der Server verwendet das Forking von Prozessen mittels fork() für die parallele Bearbeitung von Client-Anfragen und setzt sys/wait.h ein, um auf die Beendigung dieser Prozesse zu warten. Dies verhindert die Entstehung von Zombie-Prozessen, indem es sicherstellt, dass Ressourcen freigegeben werden, sobald ein Kindprozess beendet ist.

Umgang mit großen Nachrichten

Das Programm bewältigt große Datenmengen durch streambasierte Verarbeitung, zeilenweises Lesen und Schreiben sowie effiziente Pufferung. Diese Methoden ermöglichen eine reduzierte Speichernutzung und erhöhen die Leistungsfähigkeit. Zusätzlich sorgen Fehlerbehandlung für Zuverlässigkeit bei der Datenübertragung, wodurch das Programm auch umfangreiche Daten effizient verarbeitet.