Immagine che contiene simbolo, emblema, cresta, testo

Descrizione generata automaticamenteLaurea in Informatica - Università degli Studi di Salerno Corso di Ingegneria del Software - Prof. A. De Lucia

Immagine che contiene testo, logo, Carattere, Elementi grafici

Descrizione generata automaticamente

Data: 08/10/2024

TEAM

Gianluca Fusco 0512116485

Antonio Giorgio 0512106036

Cristian Di Popolo 0512105370

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 16/11/2024 | 1.0 | Introduzione e architettura software attuale | G. Fusco  A. Giorgio  C. Di Popolo |
| 18/11/2024 | 1.1 | Architettura software proposta | G. Fusco  A. Giorgio  C. Di Popolo |
| 19/11/2024 | 1.2 | Servizi dei sottosistemi | G. Fusco  A. Giorgio  C. Di Popolo |

Indice

1. Introduzione
   1. Scopo del Sistema
   2. Obiettivi e criteri di successo del progetto
   3. Definizioni, acronimi e abbreviazioni
   4. Panoramica del documento
2. **Architettura Software Attuale**
3. **Architettura Software Proposta**

3.1 Panoramica

3.2 Decomposizione in sottosistemi

3.3 Mappatura hardware/software

3.4 Gestione dei dati persistenti

3.5 Controllo degli accessi e sicurezza

3.6 Controllo globale del software

3.7 Condizioni limite

1. **Servizi dei sottosistemi** 
   1. Interfaccia Utente
   2. Gestione Autenticazione
   3. Gestione Dashboard
   4. Gestione Prenotazioni
   5. Gestione Contenuti
   6. Storage
2. INTRODUZIONE
   1. Scopo del Sistema

La nostra proposta consiste nello sviluppo di un’applicazione denominata ShopFromHome, progettata per facilitare la vendita di diverse categorie di prodotti alimentari.  
L’app offre agli utenti la possibilità di acquistare prodotti alimentari di loro scelta, supportandoli con funzionalità che consentono la gestione dei prodotti all’interno di un carrello personale.  
I gestori dell’applicazione possono visualizzare gli ordini effettuati dagli utenti, oltre a gestire l’aggiunta e la modifica dei prodotti offerti all'interno dell'app.

* 1. Obiettivi e criteri di successo del progetto

Obiettivi del progetto:

L'app ha l'obiettivo di offrire un'esperienza d'acquisto semplice e intuitiva, permettendo agli utenti di selezionare e acquistare prodotti alimentari con pochi passaggi semplici e intuitivi

Obiettivi specifici:

* Facilitare il processo di acquisto: Rendere il percorso di selezione, aggiunta al carrello e pagamento rapido, minimizzando i passaggi necessari per completare l'acquisto.
* Gestione dinamica dei prodotti: Consentire all'amministratore di aggiornare facilmente il catalogo, con la possibilità di aggiungere, modificare o rimuovere prodotti in tempo reale.
* Servizio aggiornato e rapido: Garantire che il catalogo sia sempre aggiornato, con informazioni dettagliate e corrette sui prodotti disponibili, e che le operazioni di acquisto siano veloci e sicure.

Criteri di successo:

* Esperienza utente positiva: Gli utenti devono trovare semplice e intuitivo completare l’acquisto di prodotti alimentari.
* Flessibilità e controllo per l’amministratore: L'amministratore deve poter gestire facilmente i prodotti, mantenendo un catalogo aggiornato e completo.
* Affidabilità e velocità: La piattaforma deve garantire tempi di risposta rapidi e una disponibilità continua per una user experience ottimale.
  1. Definizioni, acronimi e abbreviazioni

APP: Applicazione di acquisto online per prodotti alimentari.

ADM Amministratore: utente responsabile della gestione della piattaforma e del catalogo prodotti.

USR Utente generico: qualsiasi utilizzatore della piattaforma.

CUS   
Cliente: utente loggato che ha la possibilità di acquistare prodotti.

PROD   
Prodotto: dati e informazioni relative a un prodotto alimentare disponibile per l’acquisto.

CAT   
Catalogo: l’elenco aggiornato di prodotti disponibili sulla piattaforma.

CAR  
Carrello: spazio dove l’utente aggiunge i prodotti prima dell’acquisto.

API   
Application Programming Interface: interfaccia per l’interazione tra sistemi diversi.

UI   
User Interface: interfaccia utente che permette l’interazione con l’applicazione.

FR   
Requisiti Funzionali: caratteristiche e funzionalità essenziali dell’app.

NFR   
Requisiti Non Funzionali: caratteristiche di qualità, come performance e sicurezza.

JSON   
JavaScript Object Notation: formato di dati per lo scambio di informazioni tra client e server.

CSS   
Cascading Style Sheets: fogli di stile per la formattazione dell’interfaccia utente.

JS   
JavaScript: linguaggio di programmazione per funzioni interattive nel frontend.

CD   
Class Diagram: diagramma per la rappresentazione delle classi e delle loro relazioni.

SD   
Sequence Diagram: diagramma per visualizzare l'ordine delle operazioni tra componenti.

SCD  
StateChart Diagram: diagramma degli stati per rappresentare le diverse condizioni dell’app e le transizioni tra esse.

* 1. Panoramica del documento

In questo documento verranno mostrati i risultati del processo di progettazione del sistema. Quest’ultimo verrà descritto a livello di architettura:

* decomposizione in sottosistemi, con le loro responsabilità.
* mappatura hardware/software, descrive come i sottosistemi sono assegnati all’hardware e ai componenti di serie.
* gestione dei dati persistenti, descrivendo come sono stati memorizzati all’interno del sistema, attraverso schemi.
* controllo degli accessi e sicurezza, descrivendo l’interfaccia utente in termini di accesso, elencando i problemi di sicurezza, l’uso della crittografia e la gestione delle chiavi.
* controllo globale del software, descrive come vengono avviate le richieste e come i sottosistemi vengono sincronizzati.
* condizioni limite, descrivono l’avvio, l’arresto e il comportamento in caso di errore

1. ARCHITETTURA SOFTWARE ATTUALE

L’architettura del sistema corrente si rifà alle applicazioni dei supermercati più affermati, come l’app di conad, esselunga, coop. Le Piattaforme più diffuse, che virtualizzano il supermercato e permettono la prenotazione della spesa a distanza.

1. ARCHITETTURA SOFTWARE PROPOSTA
   1. Panoramica

Il sistema è stato progettato seguendo un’architettura Three-Tier, con l’obiettivo di

riflettere il paradigma Model-View-Controller (MVC).

Tale struttura garantisce modularità, scalabilità e facilita di manutenzione,

suddividendo il sistema in livelli con responsabilità ben definite.

L’architettura MVC e stata adottata per organizzare i sottosistemi principali come

segue:

• Model: rappresenta i sottosistemi responsabili della logica applicativa e dell’accesso

ai dati. Include le entità di dominio, la logica di business e i componenti per la

persistenza dei dati.

• View: comprende i sottosistemi che gestiscono le informazioni mostrate agli utenti,

incluse le loro interazioni con il sistema. La View riceve i dati elaborati dal Controller

e li presenta in modo chiaro e intuitivo.

• Controller: gestisce i comandi inviati dagli utenti attraverso la View. I Controller

inoltrano tali comandi al Model per eseguire operazioni, aggiornando lo stato del

sistema e restituendo i risultati alla View.

* 1. Decomposizione in sottosistemi

L'applicazione ShopFromHome è un’applicazione sviluppata seguendo

un'architettura three-tier. In questa architettura, l'interfaccia utente, la logica

applicativa e l'archiviazione dei dati sono modulari e indipendenti, garantendo

maggiore scalabilità e manutenibilità. I livelli dell'applicazione sono così suddivisi:

Presentation Layer (PL)

Questo livello rappresenta l'interfaccia utente con cui gli utenti interagiscono. Nell'app ShopFromHome, il Presentation Layer è implementato attraverso il frontend

dell'app mobile.

Gli utenti possono utilizzare attività come LoginActivity, ProductActivity e

CartActivity per navigare, effettuare operazioni di login, visualizzare prodotti,

aggiungere articoli al carrello e completare ordini. La comunicazione tra il PL e il

livello sottostante avviene tramite chiamate API REST. Il PL inoltra le richieste al

Application Layer e riceve le risposte per aggiornare l'interfaccia.

Application Layer (AL)

Questo livello intermedio si occupa di gestire la logica applicativa e di orchestrare i

flussi tra il Presentation Layer e il Data Layer.

Nell'app ShopFromHome, il backend è implementato con Spring Boot. Le richieste

provenienti dal frontend sono elaborate dai controller, che invocano i servizi per

eseguire la logica applicativa, come la gestione del carrello, la creazione di ordini o

l'autenticazione degli utenti. L'AL si occupa inoltre di inviare richieste al Data Layer

per leggere o scrivere dati persistenti.

Data Layer (DL)

Il livello più basso è responsabile della gestione dei dati persistenti.

Nell'app ShopFromHome, i dati sono archiviati in un database MySQL

secondo un modello relazionale.

Il Data Layer è integrato nel backend tramite l'uso di Spring Data JPA, che mappa le

entità applicative alle tabelle del database. Questo livello si occupa di operazioni

CRUD (Create, Read, Update, Delete) per dati relativi a utenti, prodotti, ordini e

storico degli acquisti.

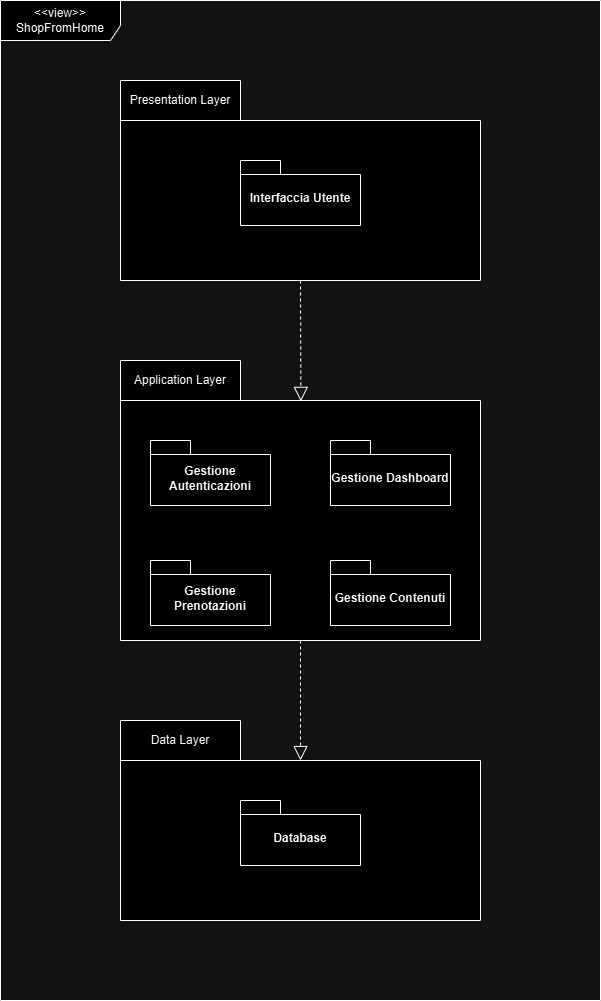
Fornisce i servizi richiesti dal Application Layer, come la ricerca di prodotti, il

recupero degli ordini di un utente o la validazione delle credenziali durante il login.

Questa suddivisione in sottosistemi consente di mantenere i livelli indipendenti,

permettendo eventuali aggiornamenti o modifiche senza impatti significativi sugli

altri strati dell'architettura.



* 1. Mappatura Hardware/Software

Il sistema ShopFromHome è un'applicazione mobile sviluppata seguendo

un'architettura client-server e accessibile tramite dispositivi mobili con

un'applicazione Android. Di seguito, vengono descritti i componenti principali del

sistema e le loro interazioni:

Client

Il lato client dell'applicazione è rappresentato dall'app mobile ShopFromHome,

sviluppata in Java, che fornisce un'interfaccia utente intuitiva basata su XML

per la struttura delle schermate e Material Design per lo stile.

La comunicazione con il backend avviene tramite chiamate API REST su protocollo

HTTPS, garantendo la sicurezza dei dati durante il transito.

Web Server

Il backend è ospitato su un server applicativo basato su Spring Boot, che funge da

web server. Il sistema utilizza un server HTTP integrato come Apache Tomcat

(incluso in Spring Boot), per gestire le richieste HTTP inviate dal client.

Database Server

Per la gestione dei dati persistenti, il sistema utilizza un database relazionale MySQL.

Il backend comunica con il database attraverso il driver JDBC, implementato tramite

il modulo Spring Data JPA. Questo consente la gestione di operazioni CRUD e query

personalizzate per entità come utenti, prodotti, carrelli e ordini.

Integrazione Client-Server

Il client invia richieste al web server utilizzando il framework Retrofit per la gestione

delle chiamate HTTP, mentre il backend restituisce i dati in formato JSON.

Infrastruttura hardware

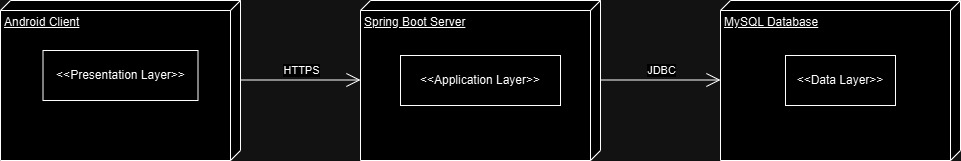
Client: Smartphone o tablet con sistema operativo Android, connesso a Internet.

Server: Un ambiente server virtuale o fisico con risorse sufficienti per ospitare Spring

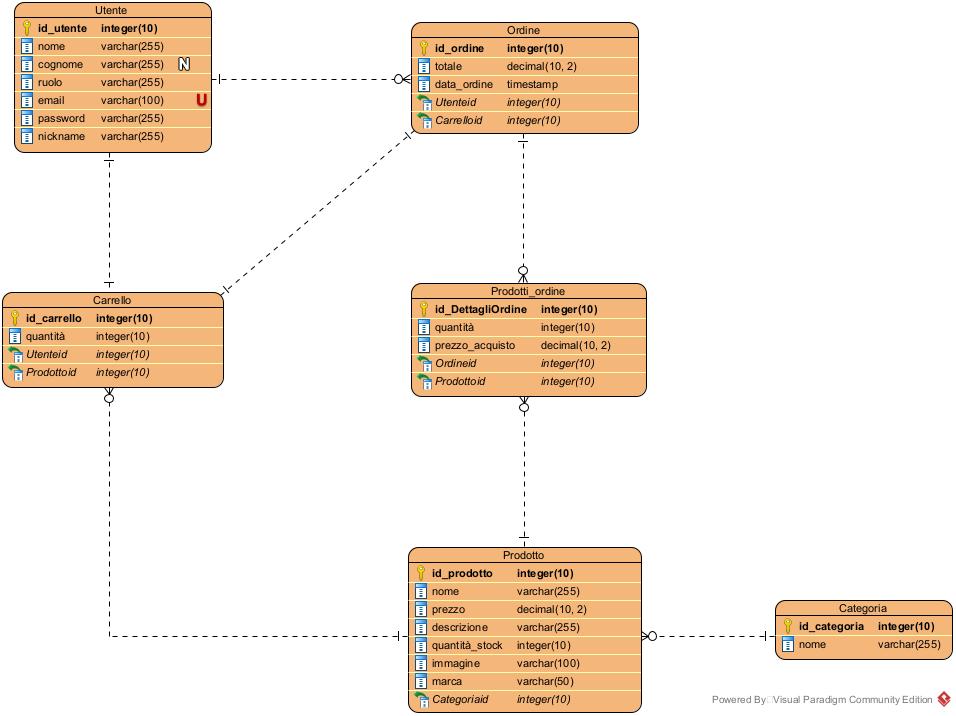
Boot e MySQL, accessibile pubblicamente su Internet tramite un indirizzo IP o un

nome di dominio.

Database Server: Installato sullo stesso server del backend.



* 1. Gestione dei dati persistenti



* 1. Controllo degli accessi e sicurezza

La comunicazione tra client e server nell'applicazione ShopFromHome avviene

tramite il protocollo di rete HTTPS, garantendo la protezione dei dati trasmessi da

eventuali intercettazioni. Le informazioni sensibili, come le credenziali di accesso,

vengono trattate con estrema sicurezza. La password degli utenti viene cifrata tramite

algoritmi di hashing, come BCrypt. Questo garantisce un ulteriore livello di

protezione, poiché le password hash non sono reversibili. L'autenticazione avviene

tramite e-mail e password. Per il controllo degli accessi, le funzionalità dell'app sono

suddivise in base alle tipologie di utenti, elencate di seguito:

Tipologie di Utenti:

Utente Ospite

Può esplorare i prodotti e aggiungerli al carrello, ma per finalizzare la creazione di un

ordine o visualizzare i dettagli dell'account è necessario registrarsi o accedere.

Cliente

Utente registrato che può effettuare ordini, gestire il proprio account e visualizzare

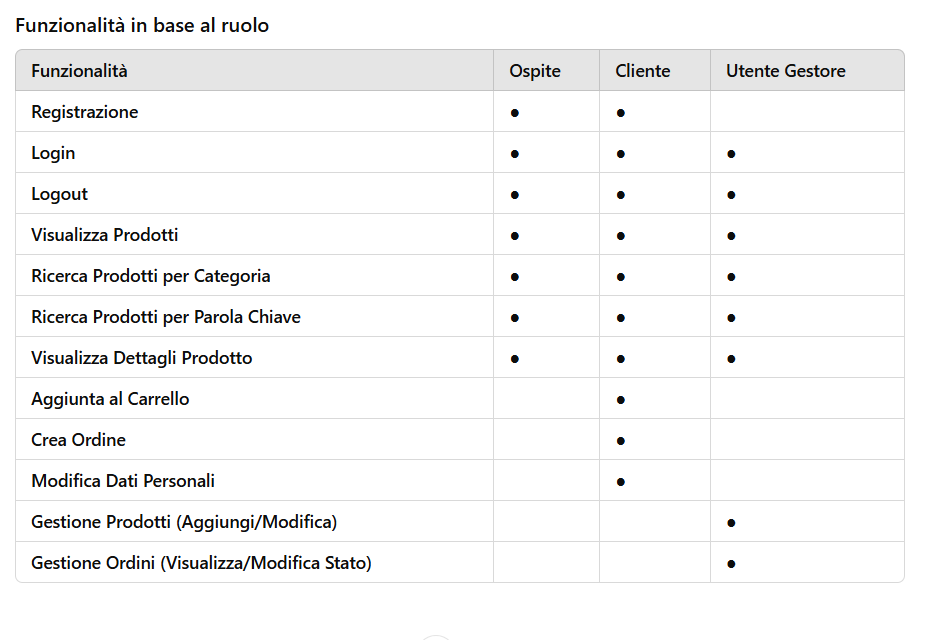
ordini.

Utente Gestore

Responsabile della gestione dei prodotti e degli ordini: può aggiungere nuovi

prodotti, modificarne i dettagli o eliminarli, oltre a monitorare e gestire gli ordini

(incluso cambiarne lo stato).



* 1. Controllo globale del software

Il controllo globale dell'app ShopFromHome è basato su un sistema di tipo event

driven control. Il ciclo principale è in ascolto in attesa di eventi esterni generati

dall'interazione degli utenti con l'interfaccia grafica. Quando un evento si verifica (ad

esempio, una richiesta di login, la selezione di un prodotto o l'aggiunta di un articolo

al carrello), il controllo viene gestito da un dispatcher nel backend.

Il dispatcher smista le richieste del client alle rispettive API REST.

Le API invocano i servizi appropriati per eseguire le operazioni richieste, come la

verifica delle credenziali, l'aggiornamento del carrello o la ricerca dei prodotti.

Dopo l'elaborazione, il backend risponde con i dati necessari per aggiornare

l'interfaccia utente tramite il frontend.

* 1. Condizioni limiti

In caso di failure dovuto a problemi hardware o software, il sistema è progettato per

garantire il ripristino dello stato precedente all'errore:

Failure del Backend

Il backend basato su Spring Boot supporta meccanismi di logging e gestione delle

eccezioni per diagnosticare e risolvere eventuali errori critici.

Failure del Database

Il DBMS MySQL implementa meccanismi di backup e ripristino per evitare la

perdita di dati importanti.

Failure del Frontend

L'app mobile conserva informazioni minime in locale per consentire agli utenti di

riprendere le attività interrotte una volta ristabilita la connessione.

1. SERVIZI DEI SOTTOSISTEMI
   1. Interfaccia Utente

L'interfaccia utente di ShopFromHome è progettata per fornire un'esperienza

semplice e intuitiva, ottimizzata per dispositivi mobili. Utilizza Android Studio per la

programmazione in Java, con XML per il layout delle schermate, Java per la logica

dell'app e JSON per la comunicazione con il backend tramite API REST. L'interfaccia

è progettata per facilitare la navigazione tra i prodotti, la gestione del carrello e l'invio

degli ordini, tenendo sempre presente la facilità d'uso.

* 1. Gestione Autenticazione

Login: Consente agli utenti di accedere al proprio account inserendo la loro email e

password. Questo sistema di autenticazione garantisce che solo gli utenti autorizzati

possano accedere all'app.

Logout: Permette agli utenti di uscire dal sistema e di rimuovere la sessione attiva.

Registrazione Cliente: Consente a un nuovo cliente di registrarsi creando un

account. L'utente può iniziare a fare acquisti una volta registrato.

* 1. Gestione Dashboard

Aggiunta Nuovo Prodotto: consente all'utente gestore di inserire un nuovo prodotto

nel catalogo. Questo include la definizione del nome del prodotto, la descrizione, il

prezzo, la categoria, la quantità disponibile, e l'immagine.

Modifica Prodotto: consente all'utente gestore di aggiornare i dettagli di un prodotto

già presente nel catalogo, come il prezzo, la descrizione, la disponibilità o

l'immagine, in caso di modifiche o aggiornamenti.

Rimozione Prodotto: consente all'utente gestore di rimuovere un prodotto dal

catalogo, rendendolo non disponibile per gli utenti.

Gestione Categorie: consente all'utente gestore di organizzare i prodotti in categorie

facilitando la navigazione e la ricerca per gli utenti.

Visualizza Prenotazioni: consente all'utente gestore di visualizzare tutte le

prenotazioni fatte dagli utenti, con dettagli su prodotti prenotati, quantità, e data

richiesta per il ritiro.

Gestione Stato Prenotazione: consente all'utente gestore di modificare lo stato di

una prenotazione (ad esempio, "In preparazione", "Pronto per il ritiro", "Completata")

per tenere traccia dell'avanzamento.

* 1. Gestione Prenotazioni

Aggiungi Prodotto al Carrello: consente all'utente di aggiungere un prodotto al

carrello per una futura prenotazione.

Rimuovi Prodotto dal Carrello: consente all'utente di rimuovere un prodotto

precedentemente aggiunto al carrello.

Modifica Quantità del Prodotto nel Carrello: consente all'utente di modificare la

quantità di un prodotto già presente nel carrello.

Crea Ordine (Checkout): consente all'utente di finalizzare la prenotazione dei

prodotti nel carrello, completando l'ordine con i dettagli necessari (come data di ritiro

e indirizzo).

* 1. Gestione Contenuti

Ricerca Prodotto per Nome: consente all'utente di cercare un prodotto nel catalogo

in base al suo nome.

Ricerca Prodotto per Categoria: consente all'utente di cercare un prodotto nel

catalogo in base alla sua categoria.

Visualizza Pagina Singolo Prodotto: consente all'utente di visualizzare la pagina

dettagliata di un prodotto selezionato, con tutte le informazioni disponibili

(descrizione, prezzo, disponibilità, etc.).

* 1. Storage

Gestire i servizi di memorizzazione per i dati persistenti relativi alle operazioni di gestione contenuti, gestione prenotazioni e gestione utenti. Questo include l'archiviazione delle informazioni sui prodotti, gli ordini, le prenotazioni, e i dati degli utenti, in modo da garantire che tutte le modifiche siano correttamente salvate e accessibili nel tempo.