Takeaway Express

Parte 1 — Spiegazione discorsiva

Takeaway Express è una web app server-side in Java che consente a clienti di effettuare ordini e allo staff di gestirli. Il cliente naviga il menu, aggiunge prodotti al carrello e conferma l'ordine tramite POST. Lo staff, dopo login, visualizza gli ordini e aggiorna il loro stato.

Il carrello è mantenuto in memoria, mentre gli ordini vengono salvati in MySQL nelle tabelle ordine e riga_ordine.

Parte 2 — Architettura tecnica

Il progetto è suddiviso in quattro package principali:

- **app** — AvvioServer e configurazione rotte
- **server** — Handler HTTP (MenuHandler, CarrelloHandler, OrdineHandler, LoginStaffHandler, DashboardStaffHandler)
- **dominio** — Logica applicativa (Carrello, Ordine, StatoOrdine, GestoreOrdini, CatalogoProdotti)
- **database** — JDBC e Query SQL

Parte 3 — Flusso HTTP

- `GET /` → Menu
- `GET /carrello?add=ID` → aggiunta prodotto
- `POST /ordine` → salvataggio ordine
- `/staff/login` → login staff
- `/staff/dashboard` → gestione ordini

Parte 4 — Database

Le tabelle principali sono:

- **prodotto** — contiene i prodotti del menu
- **ordine** — contiene la testata degli ordini
- **riga_ordine** — contiene i prodotti di ciascun ordine

Parte 5 — Sicurezza Staff

L'accesso alla dashboard è protetto tramite la classe `SessioneStaff`. Solo chi effettua login può visualizzare e modificare gli ordini.

Parte 6 — Spiegazione classe per classe

Questa sezione descrive nel dettaglio il ruolo di ogni classe del progetto e come collabora con le altre.

Package `app`

- **AvvioServer**
  È il punto di ingresso dell'applicazione. Contiene il metodo `main`. Crea l'HttpServer, inizializza gli oggetti condivisi (Carrello, CatalogoProdotti, GestoreOrdini) e registra tutte le rotte collegandole ai rispettivi Handler. In pratica è il "collante" di tutta l'applicazione.

Package `server` (livello Web)

- **MenuHandler**
  Gestisce la rotta `GET /`. Recupera i prodotti dal database tramite `CatalogoProdotti` e genera la pagina HTML del menu. Ogni prodotto ha un link che punta a `/carrello?add=ID`.
- **CarrelloHandler**
  Gestisce la rotta `GET /carrello`. Se riceve un parametro `add=ID`, cerca il prodotto e lo aggiunge al carrello, poi fa un redirect al menu. Se non ci sono parametri, mostra la pagina del carrello con righe, quantità e totale.
- **OrdineHandler**
  Gestisce `POST /ordine`. Legge i dati del form (nome, contatto, note), prende il contenuto del carrello e chiama `GestoreOrdini` per salvare l'ordine nel database. Dopo l'inserimento mostra la pagina di conferma.
- **LoginStaffHandler**
  Gestisce il login dello staff. Mostra il form (GET) e verifica le credenziali (POST) usando `SessioneStaff`. Se il login è corretto, reindirizza alla dashboard.
- **DashboardStaffHandler**
  Gestisce l'area riservata allo staff (`/staff/dashboard`). Prima di tutto controlla se l'utente è loggato. Se sì, mostra la tabella degli ordini e permette di cambiare lo stato (RICEVUTO, PRONTO, ecc.).
- **ImageHandler**
  Serve le immagini statiche (logo, foto prodotti) leggendo i file dal filesystem e inviandoli al browser con il corretto Content-Type.

Package `dominio` (logica applicativa)

- **Prodotto**
  È una classe POJO che rappresenta un prodotto del menu (id, nome, prezzo, categoria).
- **RigaCarrello**
  Rappresenta una riga del carrello: un prodotto + una quantità. Calcola il totale della riga (prezzo × quantità).
- **Carrello**
  Contiene una lista di `RigaCarrello`. Serve a memorizzare temporaneamente i prodotti scelti dal cliente prima che l'ordine venga confermato.
- **Ordine**
  Rappresenta un ordine: dati del cliente, totale, stato e data. È la "testata" dell'ordine nel database.
- **StatoOrdine** (enum)
  Definisce i possibili stati di un ordine (RICEVUTO, IN_PREPARAZIONE, PRONTO, CONSEGNATO). Serve per evitare valori non validi.
- **CatalogoProdotti**
  Classe che si occupa di leggere i prodotti dal database ed esporli come lista di oggetti `Prodotto`.
- **GestoreOrdini**
  Contiene la logica per creare ordini, salvarli nel database, leggere tutti gli ordini e aggiornare il loro stato. È il cuore della parte "gestionale".

Package `database`

- **GestoreDatabase**
  Fornisce la connessione JDBC a MySQL. Tutte le classi che parlano col database usano questa connessione.

- **Query**
  Contiene tutte le query SQL come costanti. In questo modo il codice Java rimane pulito e le query sono centralizzate.