TakeawayExpress – Documento di Analisi

Web App server-side in **Java** con `com.sun.net.httpserver.HttpServer` e database **MySQL**. HTML generato dal server. Immagini servite con handler dedicato. Regola **POST → Redirect → GET**.

Scopo del documento: descrivere in modo dettagliato (funzionale + tecnico) struttura, classi, database, rotte HTTP, flussi, tecnologie e scelte progettuali del progetto **TakeawayExpress**, da sottoporre al docente per approvazione.

Indice

1. Introduzione e obiettivi

TakeawayExpress è una web app server-side in Java che simula il processo di ordine in un locale take-away: il cliente naviga un menu diviso per categorie, aggiunge prodotti a un carrello e conferma l'ordine. Lo staff accede a un'area dedicata (con login) per visualizzare gli ordini e modificare lo stato di preparazione.

- **Obiettivo lato Cliente:** menu → carrello → conferma ordine → visualizzazione codice e stato.
- **Obiettivo lato Staff:** login → dashboard → lista ordini → dettaglio ordine → cambio stato.

2. Descrizione generale dell'applicazione

Area Cliente

- **/**: mostra il menu (categorie: antipasti/panini/bevande) e permette di aggiungere prodotti.
- **/carrello**: riepilogo ordine con pulsanti **Aggiungi altro** e **Conferma ordine**.
- **/ordine**: conferma con codice (es. `T-00012`).
- **/ordine/stato**: tracking stato ordine.

Area Staff

- **/admin/login**: autenticazione semplice (password/identificativo).
- **/admin**: dashboard con scelta: **Menu** o **Ordini**.
- **/admin/ordini**: tabella ordini.
- **/admin/ordine**: dettaglio ordine e cambio stato.

3. Architettura del sistema

L'architettura segue una separazione per responsabilità: **handler HTTP** (routing e request/response), **dominio** (logica business), **database** (JDBC + Query centralizzate), **pagine** (HTML server-side) e **util** (helper).

```
Client (Browser) → HttpServer (Handlers) → (Dominio + Database) → Pagine (HTML) → Response
```

Nota: HTML viene prodotto dal server (nessun framework esterno). Le immagini vengono servite tramite endpoint statico `/img`.

4. Struttura del progetto

4.1 Package app (Avvio applicazione)

Classe	Tipo	Descrizione
AvvioServer	Classe main	Punto di ingresso dell'applicazione. Avvia l'HttpServer, inizializza database, catalogo, carrello e gestore ordini e registra tutte le rotte HTTP.

4.2 Package `dominio` (modello e logica)

Classe	Tipo	Ruolo
Prodotto	Classe modello	Rappresenta un prodotto del menu (id, nome, descrizione, prezzo, categoria).
RigaCarrello	Classe modello	Rappresenta una riga del carrello (prodotto + quantità).
Carrello	Classe di servizio	Gestisce una collezione di righe e calcola il totale dell'ordine.
Ordine	Classe modello	Rappresenta un ordine confermato con codice, cliente, totale e stato.
StatoOrdine	Enum	Definisce gli stati dell'ordine: RICEVUTO, IN_PREPARAZIONE, PRONTO, CONSEGNATO.
CatalogoProdotti	Classe di servizio	Carica i prodotti dal database e li rende disponibili al menu.
GestoreOrdini	Classe business	Crea ordini dal carrello, li salva nel database e ne gestisce lo stato.

4.3 Package `database`

Classe	Tipo	Ruolo
GestoreDatabase	Classe di infrastruttura	Gestisce la connessione JDBC verso MySQL.
Query	Classe utility	Contiene tutte le query SQL usate dall'applicazione.

4.4 Package `server` (HTTP layer)

Classe	Area	Funzione
MenuHandler	Cliente	Mostra il menu principale.
CarrelloHandler	Cliente	Gestisce aggiunta, modifica e visualizzazione del carrello.
OrdineHandler	Cliente	Conferma l'ordine e mostra lo stato.
LoginStaffHandler	Staff	Gestisce il login del personale.
DashboardStaffHandler	Staff	Pagina iniziale dopo il login.
ListaOrdiniStaffHandler	Staff	Mostra la tabella con tutti gli ordini.
DettaglioOrdineStaffHandler	Staff	Mostra il dettaglio di un ordine e consente di cambiarne lo stato.
MenuStaffHandler	Staff	Consente allo staff di visualizzare il menu.
ImageHandler	Statico	Serve le immagini dei prodotti dal filesystem.

4.5 Package `pagine` (HTML server-side)

Classe	Tipo	Ruolo
LayoutPagina	Classe base	Fornisce header e footer comuni a tutte le pagine.
PaginaMenu	Generatore HTML	Pagina del menu cliente.
PaginaCarrello	Generatore HTML	Pagina del carrello.
PaginaOrdine	Generatore HTML	Conferma ordine.
PaginaStatoOrdine	Generatore HTML	Tracking stato ordine.
PaginaLoginStaff	Generatore HTML	Pagina login staff.
PaginaDashboardStaff	Generatore HTML	Dashboard staff.
PaginaListaOrdiniStaff	Generatore HTML	Lista ordini.
PaginaDettaglioOrdineStaff	Generatore HTML	Dettaglio ordine.
PaginaMenuStaff	Generatore HTML	Menu per lo staff.

4.6 Package `util`

Classe	Tipo	Ruolo
HttpUtils	Utility	Gestione risposte HTTP, redirect e status code.
HtmlUtils	Utility	Funzioni di supporto per la generazione dell'HTML.

5. Descrizione dettagliata delle classi

5.1 Avvio e routing

Classe	Responsabilità	Note/Variabili principali
AvvioServer	Avvia `HttpServer`, registra le rotte (`createContext`), imposta porta e executor.	• porta (es. 8080) • istanze condivise: `CatalogoProdotti`, `Carrello`, `GestoreOrdini`

5.2 Dominio (logica business)

Classe	Responsabilità	Attributi principali
Prodotto	Rappresenta un prodotto del menu.	`id`, `nome`, `descrizione`, `prezzo`, `categoria`
RigaCarrello	Rappresenta una riga del carrello (prodotto + quantità) e calcola subtotale.	`Prodotto prodotto`, `int quantita`
Carrello	Contiene le righe selezionate dal cliente. Permette aggiunta/aggiornamento/rimozione e calcolo totale.	• Struttura consigliata: `Map<String, RigaCarrello>` (chiave = id prodotto) • Metodi: `aggiungi()`, `aggiorna()`, `svuota()`, `getTotale()`, `getTotaleQuantita()`
Ordine	Rappresenta un ordine confermato.	`codice`, `nomeCliente`, `contatto`, `note`, `totale`, `StatoOrdine`, date
StatoOrdine	Enum degli stati ordine e regole di avanzamento.	`RICEVUTO` → `IN_PREPARAZIONE` → `PRONTO` → `CONSEGNATO`

CatalogoProdotti	Carica e fornisce la lista prodotti (dal DB). Supporta filtri per categoria.	`List<Prodotto> prodotti` + metodi `getPerCategoria()`
GestoreOrdini	Servizio principale: crea ordine dal carrello, salva su DB, recupera ordini, aggiorna stato.	• Dipende da `GestoreDatabase` • Metodi: `creaOrdineDaCarrello()`, `getOrdini()`, `getDettaglio()`, `cambiaStato()`

5.3 Database (JDBC + Query)

Classe	Responsabilità	Note
GestoreDatabase	Gestione connessione MySQL (URL, utente, password). Espone metodi per ottenere `Connection`.	Uso di `PreparedStatement` e chiusura risorse (try-with-resources).
Query	Centralizza tutte le query SQL (prodotti, ordini, righe ordine).	Evita query sparse nel progetto, migliora manutenzione e coerenza.

5.4 Server (Handlers HTTP)

Handler	Rotte	Responsabilità
MenuHandler	GET `/`	Carica prodotti dal catalogo e richiama `PaginaMenu`.
CarrelloHandler	GET `/carrello` + POST `/carrello/aggiungi`, `/carrello/aggiorna`, `/carrello/svuota`	Gestisce modifiche al carrello e applica PRG (redirect).
OrdineHandler	POST `/checkout/conferma`, GET `/ordine`, GET `/ordine/stato`	Crea ordine su DB, mostra conferma e tracking.
LoginStaffHandler	GET/POST `/admin/login`	Gestisce login staff (controllo credenziali).
DashboardStaffHandler	GET `/admin`	Mostra scelta iniziale staff: Menu / Ordini.
ListaOrdiniStaffHandler	GET `/admin/ordini`	Visualizza tabella ordini, filtri opzionali.
DettaglioOrdineStaffHandler	GET `/admin/ordine` + POST `/admin/ordine/stato`	Dettaglio ordine e cambio stato con validazione.
MenuStaffHandler	GET `/admin/menu`	Mostra menu allo staff (consultazione).
ImageHandler	GET `/img/*`	Serve file statici (PNG/JPG) dalla cartella `./img`.

5.5 Pagine (HTML server-side)

Le classi in `pagine/` si occupano solo della generazione HTML. Gli handler chiamano le pagine passando i dati (lista prodotti, carrello, ordini).
Regola: niente HTML dentro gli handler, per separare logica HTTP e presentazione.

Classe	Responsabilità
LayoutPagina	Header comune (nome locale, link carrello con contatore, link staff) + footer.
PaginaMenu	Elenco prodotti per categoria con form per aggiungere al carrello.
PaginaCarrello	Riepilogo righe, totale, pulsanti **Aggiungi altro** e **Conferma ordine**.
PaginaOrdine	Conferma ordine (codice ordine, riepilogo).

PaginaStatoOrdine	Tracking stato ordine (stato attuale, data aggiornamento).
PaginaLoginStaff	Form login staff.
PaginaDashboardStaff	Schermata iniziale staff: pulsanti Menu / Ordini.
PaginaListaOrdiniStaff	Tabella ordini con link al dettaglio.
PaginaDettaglioOrdineStaff	Dettaglio ordine + pulsanti cambio stato (in base allo stato attuale).
PaginaMenuStaff	Visualizzazione menu per consultazione staff.

6. Database: schema, tabelle e relazioni

Il database viene creato e gestito in MySQL Workbench. Lo schema include: **prodotto** (menu), **ordine** (testata) e **riga_ordine** (dettaglio).

6.1 Tabella `prodotto`

Campo	Tipo	Vincoli	Descrizione
id	VARCHAR(20)	PK, NOT NULL	ID prodotto (usato anche per immagine: `/img/<id>.png`).
nome	VARCHAR(100)	NOT NULL	Nome prodotto.
descrizione	TEXT	NULL	Descrizione breve.
prezzo	DECIMAL(10,2)	NOT NULL, CHECK >= 0	Prezzo di vendita.
categoria	VARCHAR(50)	NOT NULL	Categoria (Antipasto/Panino/Bevanda).

6.2 Tabella `ordine`

Campo	Tipo	Vincoli	Descrizione
id	INT	PK, AUTO_INCREMENT	Identificativo interno.
codice	VARCHAR(20)	UNIQUE, NOT NULL	Codice pubblico ordine (es. `T-00012`).
nome_cliente	VARCHAR(100)	NOT NULL	Nome cliente.
contatto	VARCHAR(100)	NOT NULL	Telefono/email.
note	TEXT	NULL	Note opzionali.
totale	DECIMAL(10,2)	NOT NULL, CHECK >= 0	Totale ordine.
stato	VARCHAR(30)	NOT NULL	Stato ordine (coerente con `StatoOrdine`).
data_creazione	DATETIME	NOT NULL	Timestamp creazione.
data_aggiornamento	DATETIME	NULL	Ultimo aggiornamento stato (tracking).

6.3 Tabella `riga_ordine`

Campo	Tipo	Vincoli	Descrizione
id	INT	PK, AUTO_INCREMENT	ID riga.
ordine_id	INT	FK → ordine(id), NOT NULL	Collegamento alla testata ordine.
prodotto_id	VARCHAR(20)	FK → prodotto(id), NOT NULL	Prodotto ordinato.
quantita	INT	NOT NULL, CHECK > 0	Numero pezzi.
prezzo_unitario	DECIMAL(10,2)	NOT NULL, CHECK >= 0	Prezzo al momento dell'ordine (storico).

6.4 Relazioni

- **ordine (1) → (N) riga_ordine**: un ordine contiene più prodotti.
- **prodotto (1) → (N) riga_ordine**: lo stesso prodotto può comparire in ordini diversi.

7. Rotte HTTP e parametri

Le rotte sono progettate per rispettare la regola: **tutte le modifiche (scritture) avvengono in POST** e poi si effettua un redirect verso una GET (PRG).

7.1 Rotte Cliente

Metodo	URL	Parametri	Descrizione	Esito
GET	`/`	-	Menu prodotti (con filtri categoria lato UI).	200 HTML
POST	`/carrello/aggiungi`	`prodotto`, `quantita`	Aggiunge/aggiorna una riga nel carrello.	302 → `/carrello`
GET	`/carrello`	-	Riepilogo carrello.	200 HTML
POST	`/carrello/aggiorna`	`q_` ...	Aggiorna quantità per ogni riga.	302 → `/carrello`
POST	`/carrello/svuota`	-	Svuota carrello.	302 → `/carrello` o `/`
POST	`/checkout/conferma`	`nome`, `contatto`, `note` (opz)	Crea ordine su DB (testata + righe), svuota carrello.	302 → `/ordine?codice=...`
GET	`/ordine`	`codice`	Conferma ordine e riepilogo.	200 HTML
GET	`/ordine/stato`	`codice` (opz)	Tracking stato ordine.	200 HTML

7.2 Rotte Staff

Metodo	URL	Parametri	Descrizione	Esito
GET	`/admin/login`	-	Pagina login staff.	200 HTML
POST	`/admin/login`	`identificativo`, `password`	Verifica credenziali.	302 → `/admin` (o errore)
GET	`/admin`	-	Dashboard staff (Menu / Ordini).	200 HTML
GET	`/admin/ordini`	`stato` (opz)	Tabella ordini, filtro opzionale.	200 HTML
GET	`/admin/ordine`	`codice`	Dettaglio ordine e pulsanti cambio stato.	200 HTML
POST	`/admin/ordine/stato`	`codice`, `stato`	Applica transizione stato se valida.	302 → `/admin/ordine?codice=...`
GET	`/admin/menu`	-	Visualizzazione menu per consultazione staff.	200 HTML

8. Gestione immagini statiche

Le immagini non vengono salvate nel database. Vengono gestite come risorse statiche nella cartella `./img` e servite tramite `ImageHandler` sulla rotta `/img`.

- Convenzione: per un prodotto con `id=CBO` l'immagine è `/img/CBO.png`.
- Il browser richiede l'immagine e il server risponde con `Content-Type` corretto (png/jpg).

9. Tecnologie e librerie utilizzate

Componente	Libreria/Package	Utilizzo
HTTP Server	`com.sun.net.httpserver`	Routing e gestione richieste.
JDBC	`java.sql`	Connessione MySQL, query con PreparedStatement.
File system	`java.nio.file`	Lettura file immagini e content-type.
Time	`java.time`	Gestione date/ore (opzionale lato Java; lato DB si usa DATETIME).

10. Variabili e oggetti condivisi

Per semplicità didattica (prima versione), il progetto mantiene alcune istanze condivise create in `AvvioServer` e passate ai handler:

- `GestoreDatabase db` : connessione/creazione `Connection` .
- `CatalogoProdotti catalogo` : prodotti disponibili, caricati da DB.
- `Carrello carrello` : carrello corrente (nota: senza sessioni avanzate, è condiviso; per demo è accettabile).
- `GestoreOrdini gestoreOrdini` : crea ordini e cambia stati.

11. Validazioni e gestione errori

11.1 Validazioni principali

- **Quantità**: deve essere intero > 0 (o >=0 in aggiornamento).
- **Prodotto**: l'ID prodotto deve esistere nel catalogo.
- **Checkout**: nome e contatto obbligatori.
- **Codice ordine**: su pagine ordine/staff deve esistere, altrimenti errore 404 o pagina "Ordine non trovato".
- **Stato ordine**: transizione valida senza salti (es. RICEVUTO → PRONTO non consentito).

11.2 Error handling

- Parametri mancanti: risposta con pagina errore chiara (400) o messaggio in pagina.
- Risorsa non trovata: 404 (es. immagine assente, ordine inesistente).
- Metodo non consentito: 405 (se una rotta riceve un metodo non previsto).
- Eccezioni DB: log server-side e pagina errore generica per utente.

12. Scelte progettuali e motivazioni

- **Separazione Handler / Pagine**: gli handler gestiscono solo HTTP; le pagine generano HTML → maggiore manutenibilità.
- **Query centralizzate**: tutte le SQL in `Query` → coerenza e facilità di modifica.
- **PRG (POST→Redirect→GET)**: evita doppio invio form con refresh e migliora UX.
- **Immagini fuori dal DB**: semplifica il database e rende le risorse gestibili come file statici.
- **Stati ordine come enum**: riduce errori e rende esplicite le transizioni consentite.

Deliverable dimostrativi (demo):

- Catalogo con almeno 6 prodotti in 3 categorie.
- Creazione di almeno 2 ordini da area cliente.
- Gestione e avanzamento stato ordini da area staff (almeno 2 cambi stato).