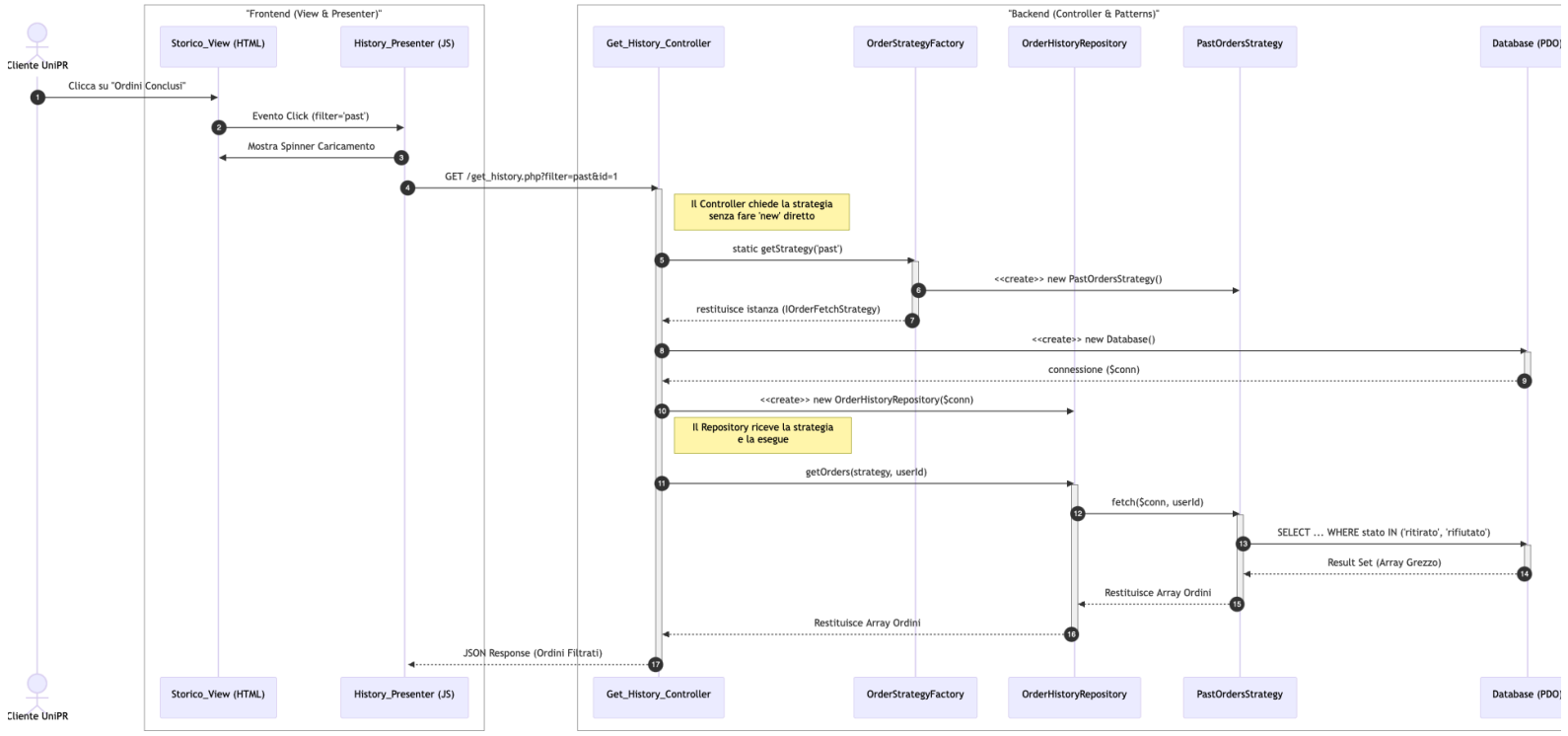
SVOLTO DA COLUCCI PASQUALE, MATR: 358141

Diagramma di Sequenza: Modulo Storico Ordini



Descrizione del Diagramma di Sequenza: Modulo Storico Ordini

Il diagramma di sequenza modella il comportamento dinamico del sistema nello scenario in cui l'utente richiede la visualizzazione di una specifica categoria di ordini (es. "Ordini Conclusi"). Il flusso evidenzia l'interazione asincrona tra i livelli architetturali e l'applicazione dei pattern comportamentali.

1. **Attivazione e Gestione Asincrona (Frontend MVP):** L'interazione è innescata dall'utente sulla **View** (`Storico.html`). L'evento viene catturato dal **Presenter** (`History_Presenter`), che impedisce il ricaricamento della pagina e avvia una richiesta AJAX verso il server. Questo approccio garantisce un'esperienza utente fluida e reattiva (feedback immediato di caricamento).
2. **Creazione Dinamica della Strategia (Factory Pattern):** Il **Controller** (`Get_History_Controller`) riceve la richiesta HTTP con il parametro di filtro. Invece di contenere logica condizionale per scegliere la query, il Controller delega questa decisione alla classe **`OrderStrategyFactory`**.

    - La Factory istanzia e restituisce l'oggetto strategia corretto (es. `PastOrdersStrategy`) incapsulato nell'interfaccia generica `IOrderFetchStrategy`.
3. **Iniezione delle Dipendenze (Dependency Injection):** Il Controller prepara l'ambiente di esecuzione istanziando la connessione al **Database** e iniettandola nel costruttore del **Repository** (`OrderHistoryRepository`). Successivamente, passa la strategia ottenuta al Repository tramite il metodo `getOrders`.
4. **Esecuzione Polimorfica (Strategy Pattern):** Il Repository agisce come *Context*: non conosce i dettagli della query SQL che sta per eseguire. Si limita a invocare il metodo `fetch($conn, $userId)` sull'oggetto strategia ricevuto.
    - Sarà la classe concreta (`PastOrdersStrategy`) a eseguire la logica specifica (es. `SELECT ... WHERE stato = 'ritirato'`) utilizzando la connessione ricevuta.
5. **Risposta e Rendering:** I dati grezzi recuperati dal database risalgono la catena fino al Controller, che li restituisce al client in formato JSON standardizzato. Il **Presenter** elabora questi dati (formattazione date, selezione colori badge) e aggiorna puntualmente il DOM della View, completando il ciclo MVP.

Nota sull'Ottimizzazione del Dettaglio Ordini: Per minimizzare il numero di richieste HTTP verso il server e rispettare l'approccio stateless, l'architettura adotta una strategia di **Eager Loading** per i dettagli dell'ordine. Le classi `Strategy` recuperano già, tramite aggregazione SQL (`GROUP_CONCAT`), la sintesi dei prodotti ordinati (campo `dettagli`). Di conseguenza, l'interazione utente "Espandi Dettaglio Ordine" viene gestita interamente lato client dal **Presenter** senza richiedere una nuova interazione nel Diagramma di Sequenza, migliorando la reattività dell'applicazione.