**3. Progettazione e Sviluppo Iterazione 2**

**3.1. Progettazione e Sviluppo Caso d’uso Update Stato**

Nella seconda iterazione del progetto, è stato implementato il caso d'uso UC5: Update Stato. Questo caso d'uso descrive come il corriere aggiorna lo stato di una spedizione utilizzando il design pattern Observer.

Il processo di aggiornamento dello stato avviene come segue:

1. Il corriere chiama il metodo updateStato() per aggiornare lo stato della spedizione.
2. Utilizzando il design pattern Observer, il corriere registra la spedizione come suo osservatore.
3. Il corriere chiama il metodo getCodiceQR() della spedizione per ottenere il codice QR associato.
4. Successivamente, esegue il decrypt del codice QR per ottenere in chiaro le informazioni, tra cui lo stato della spedizione. Vengono rimossi i primi 42 caratteri della stringa, che corrispondono a 16 caratteri del codice fiscale del mittente, 16 caratteri del codice fiscale del destinatario e 10 caratteri della data di consegna nel formato "yyyy-MM-dd".
5. In base allo stato corrente della spedizione, il corriere determina il nuovo stato secondo la seguente logica:
   1. Se lo stato della spedizione è "Nuovo", il nuovo stato viene impostato come "Spedito".
   2. Se lo stato della spedizione è "Spedito", il nuovo stato viene impostato come "In deposito".
   3. Se lo stato della spedizione è "In deposito", il nuovo stato viene impostato come "In Consegna".
   4. Se lo stato della spedizione è "In Consegna", il nuovo stato viene impostato come "Consegnato".
6. Il corriere il metodo updateStato(nuovoStato) della spedizione, passando il nuovo stato come parametro. Questo metodo imposta il nuovo stato della spedizione.
7. Successivamente, viene chiamato il metodo updateCodiceQR() della spedizione per aggiornare il codice QR associato alla spedizione. Viene criptata una stringa contenente il codice fiscale del mittente, il codice fiscale del destinatario, la data di spedizione e lo stato della spedizione.

La spedizione è stata aggiornata con il nuovo stato e il nuovo codice QR associato.

A picture containing text, screenshot, diagram, font

Description automatically generated

**3.2. Implementazione Classi CRUD con Pattern DAO**

Di seguito viene descritto come sono state implementate le classi CRUD DAO per le entità Cliente (ClienteDAO), Corriere (CorriereDAO), Spedizione (SpedizioneDAO) e Magazzino (MagazzinoDAO):

**ClienteDAO**

La classe ClienteDAO fornisce le operazioni di CRUD (Create, Read, Update, Delete) per l'entità Cliente.

**Metodi principali:**

* createCliente(Cliente cliente): Crea un nuovo record nel database per il cliente specificato.
* getClienteById(int id): Recupera il cliente dal database utilizzando l'ID specificato.
* updateCliente(Cliente cliente): Aggiorna le informazioni del cliente nel database.
* deleteCliente(Cliente cliente): Elimina il cliente specificato dal database.

**CorriereDAO**

La classe CorriereDAO gestisce le operazioni di CRUD per l'entità Corriere.

**Metodi principali:**

* createCorriere(Corriere corriere): Crea un nuovo record nel database per il corriere specificato.
* getCorriereById(int id): Recupera il corriere dal database utilizzando l'ID specificato.
* updateCorriere(Corriere corriere): Aggiorna le informazioni del corriere nel database.
* deleteCorriere(Corriere corriere): Elimina il corriere specificato dal database.

Stessa logica è stata utilizzata anche per implementazione MagazzinoDAO e SpedizioneDAO.

Le classi CRUD DAO sopra descritte seguono il pattern DAO (Data Access Object) che separa la logica di accesso ai dati dal resto dell'applicazione. Questo consente di gestire facilmente le operazioni di persistenza delle entità nel database e offre un'interfaccia chiara e standardizzata per eseguire le operazioni CRUD.