



Università degli Studi di Salerno

Corso di Ingegneria del Software



System Design Document

Partecipanti:

Nome	Matricola
Carlo Lerro	0512105440
Alfonso Fiorentino	0512102994
Antonio Botticchio	0512105458

Storico Revisioni

<i>Data</i>	<i>Versione</i>	<i>Cambiamenti</i>	<i>Autori</i>
11/11/2019	0.1	Aggiunta sezione 1	Carlo Lerro
15/11/2019	0.2	Aggiunta sezione 2	Carlo Lerro, Alfonso Fiorentino
22/11/2019	0.3	Stesura sezione 3	Carlo Lerro
29/11/2019	0.4	Aggiunta "Decomposizione in sottosistemi"	Carlo Lerro
06/12/2019	0.5	Aggiunta matrice degli accessi e modifica dei sottosistemi	Carlo Lerro
07/12/2019	0.6	Aggiunti scenari boundary	Carlo Lerro

		condition	
08/12/2019	0.7	Aggiunta sezione 4	Carlo Lerro
17/12/2019	0.8	Aggiunta sezione 3.4	Carlo Lerro
16/01/2020	0.9	Revisione Documento	Alfonso Fiorentino
18/01/2020	1.0	Revisione Documento	Carlo Lerro

Introduction	3
Purpose of the system	3
Design goals	3
Current software architecture	4
Analisi delle piattaforme presenti	4
Proposed system architecture	6
Overview	6
Subsystem decomposition	7
Decomposizione in Layer	7
Decomposizione in sottosistemi	8
Hardware/software mapping	8
Persistent data management	9
Access control and security	9
Global software control	9
Boundary conditions	10
Subsystem services	12

1. Introduction

1.1. Purpose of the system

Il sistema che si vuole realizzare ha come scopo la facilitazione della gestione degli ordini delle pizzerie, ovvero il monitoraggio degli stessi.

L'obiettivo è di realizzare un sistema che permetta di facilitare il lavoro dei ristoratori, ma non solo, anche quello dei fattorini. Inoltre è importante facilitare l'ordinazione da parte del cliente, quindi creando un sistema semplice ed intuitivo.

Per raggiungere tale obiettivo ci si è posti dalla parte, sia del ristoratore, sia del fattorino, e sia del cliente.

Il sistema progettato è una vera e propria web app a cui avranno accesso: i clienti, i ristoratori, il gestore account, ed i fattorini.

1.2. Design goals

- **Criteri di performance**

- Tempi di risposta: il sistema dovrà consentire una navigazione fluida ai vari utenti, quindi con tempi di risposta bassi nello svolgimento delle funzionalità offerte dal sistema

- **Criteri di affidabilità**

- Robustezza: il sistema informerà l'utente di eventuali errori nel caso di immissione di input non validi attraverso degli appositi messaggi
- Affidabilità: il sistema deve garantire l'affidabilità dei servizi proposti. Il sistema dovrà essere in grado di controllare accuratamente ogni input inserito dall'utente per garantire l'affidabilità
- Disponibilità: il sistema dovrà essere disponibile ogni qual volta un utente ne richiede l'utilizzo
- Sicurezza: l'accesso al sistema avviene mediante e-mail e password. Inoltre, la sicurezza è garantita in quanto ogni utente può svolgere solo le operazioni ad esso assegnate.

- **Criteri di manutenzione**

- Estensibilità: è possibile aggiungere nuove funzionalità al sistema in base alle esigenze degli utenti o all'introduzione di nuove tecnologie
- Portabilità: l'interazione con il sistema avviene mediante un browser, quindi il sistema può essere definito portatile in quanto può funzionare su qualsiasi dispositivo mobile o desktop, che

esso sia, affinché si utilizzi un browser più o meno recente, compatibile con le nuove tecnologie.

- **Criteri di usabilità**

- Usabilità: il sistema sarà di facile comprensione ed utilizzo, permettendo di effettuare in modo semplice ed immediato le varie operazioni grazie ad un'interfaccia user-friendly
- Utilità: Il sistema è utile in quanto si migliora e velocizza il processo di ordinazione e consegna degli ordini delle pizzerie.

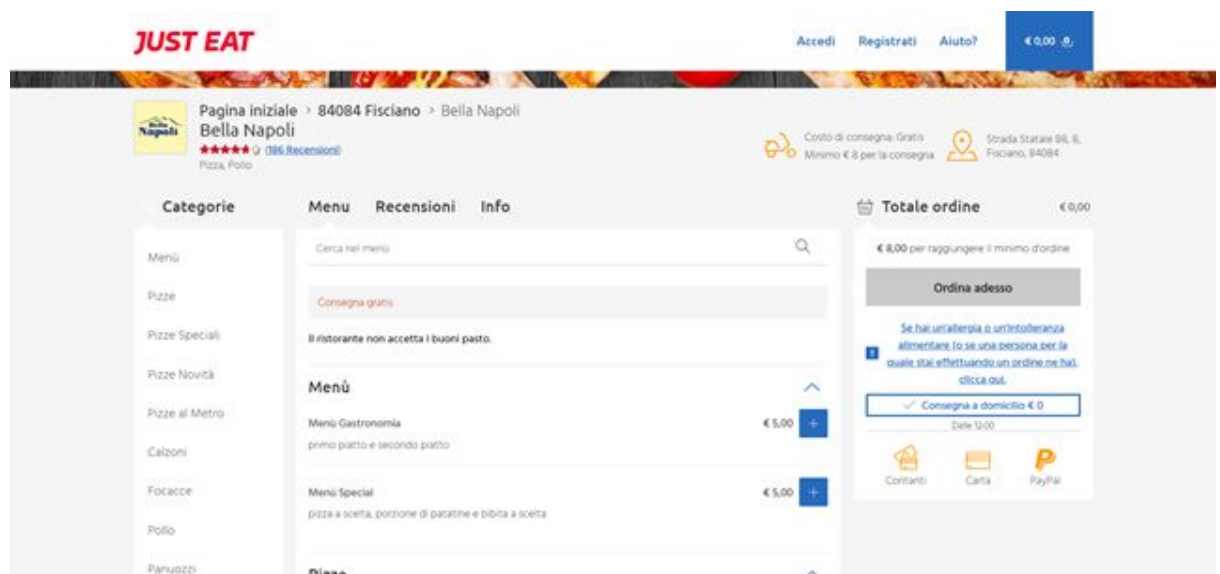
2. Current software architecture

Attualmente esistono piattaforme che consentono la possibilità di prenotare comodamente da casa.

2.1 Analisi delle piattaforme presenti

Just Eat:

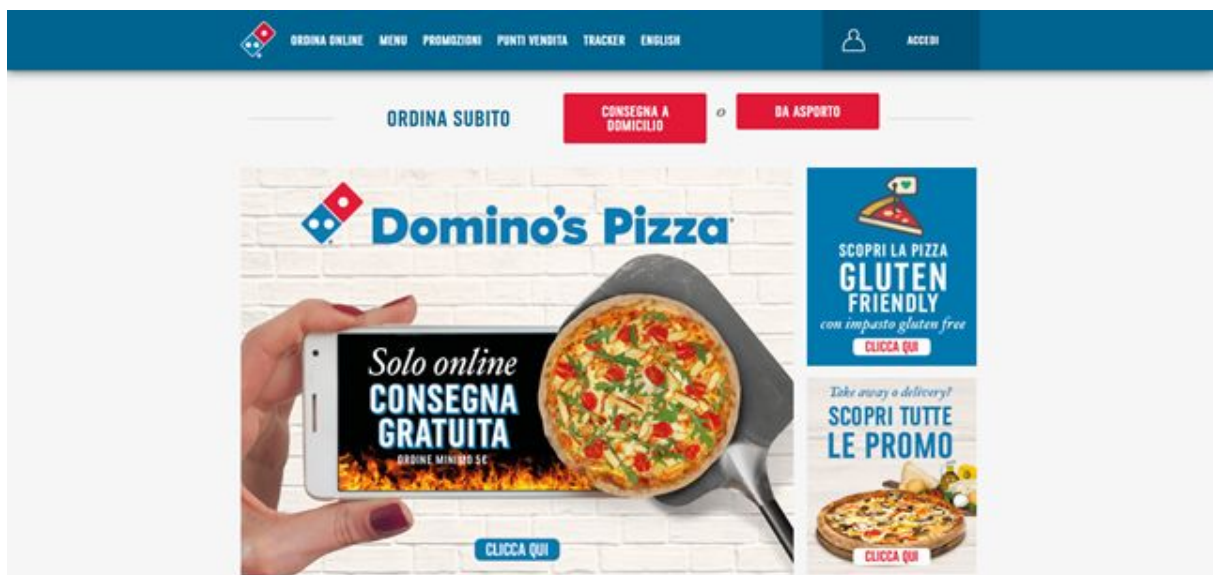
Seppur più generico rispetto a ciò che vorremmo realizzare, in quanto non tratta solo pizze ma cibo in generale, rappresenta il miglior competitor esistente nel mercato; a tal proposito abbiamo deciso di analizzare il sito per poter progettare una piattaforma quanto più efficiente possibile. Esso è strutturato in:



- Ordina: inserire l'indirizzo dove effettuare la consegna, ed una volta scelta la pizzeria

- Menù: i cibi acquistabili
- Recensioni: le valutazioni della pizzeria
- Info: informazioni generiche (orari, piccola descrizione)

Domino's Pizza:



Molto meno famoso del precedente, è la piattaforma che più si avvicina alla nostra. Essa difatti è un franchising, molto simile a ciò che offrirà la nostra piattaforma. Il sito presenta le seguenti categorie:

- Ordina Online: permette di effettuare l'ordine della pizza online
- Menu: informazioni riguardo le pizze offerte ed altri prodotti ordinabili
- Promozioni: le promozioni del giorno / settimana
- Punti vendita: informazioni riguardo le pizzerie affiliate al sito
- Tracker: controllo dello stato di avanzamento dell'ordine

Notiamo però in esso l'impossibilità da parte dell'utente di conoscere, o di avere ulteriori informazioni, riguardo la pizzeria presso la quale sta effettuando l'ordine.

Entrambi i siti esistenti permettono l'iscrizione alla piattaforma per poter effettuare velocemente i successivi ordini e per ricordare le preferenze dell'utente; nonché un pannello di controllo per ciascuna pizzeria / ristorante affiliato.

3. Proposed system architecture

3.1. Overview

Il sistema da noi proposto è un'applicazione web che gira su un server. Gli utenti come detto, saranno: Clienti, Ristoratori, Gestore Utenti e Fattorino. Tutti gli utenti potranno fare login-logout, i clienti potranno registrarsi grazie al form di login, mentre i ristoratori inseriranno i propri dati per la registrazione al momento della compilazione della richiesta di affiliazione e, in un successivo momento, se la proposta, verrà accettata dal gestore utenti, sarà registrato in automatico dal sistema. I fattorini invece, verranno registrati dal gestore utenti. Ogni tipologia di utente avrà le proprie funzionalità. In particolare il cliente potrà effettuare ordini, questi in automatico verranno inoltrati al ristoratore, che deciderà a quale fattorino affidare l'ordine, quest'ultimo verrà notificato dell'avvenuto affidamento di un ordine.

Inoltre i ristoratori potranno inviare le richieste che, prese in carico dal gestore utenti deciderà se accogliere o meno la proposta.

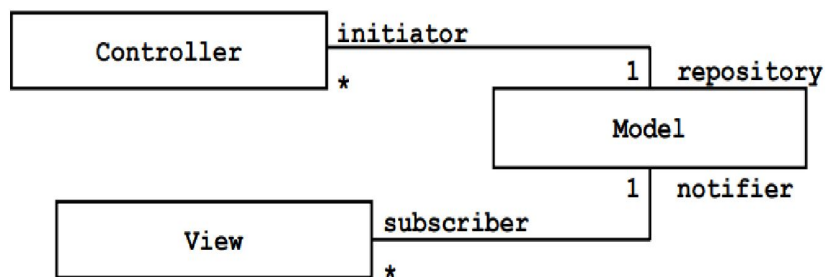
L'architettura scelta è di tipo repository, poichè sono adatti per applicazioni con task di elaborazione dati che cambiano di frequente, nello specifico è un sistema MVC. Quest'ultimo è un pattern architetturale molto diffuso nello sviluppo di interfacce grafiche di sistemi software object-oriented, in grado di separare la logica di presentazione dei dati, dalla logica di business. E' un'architettura multi-tier, dove le varie funzionalità del sito sono logicamente separate e suddivise su più strati o livelli software differenti in comunicazione tra loro.

3.2. Subsystem decomposition

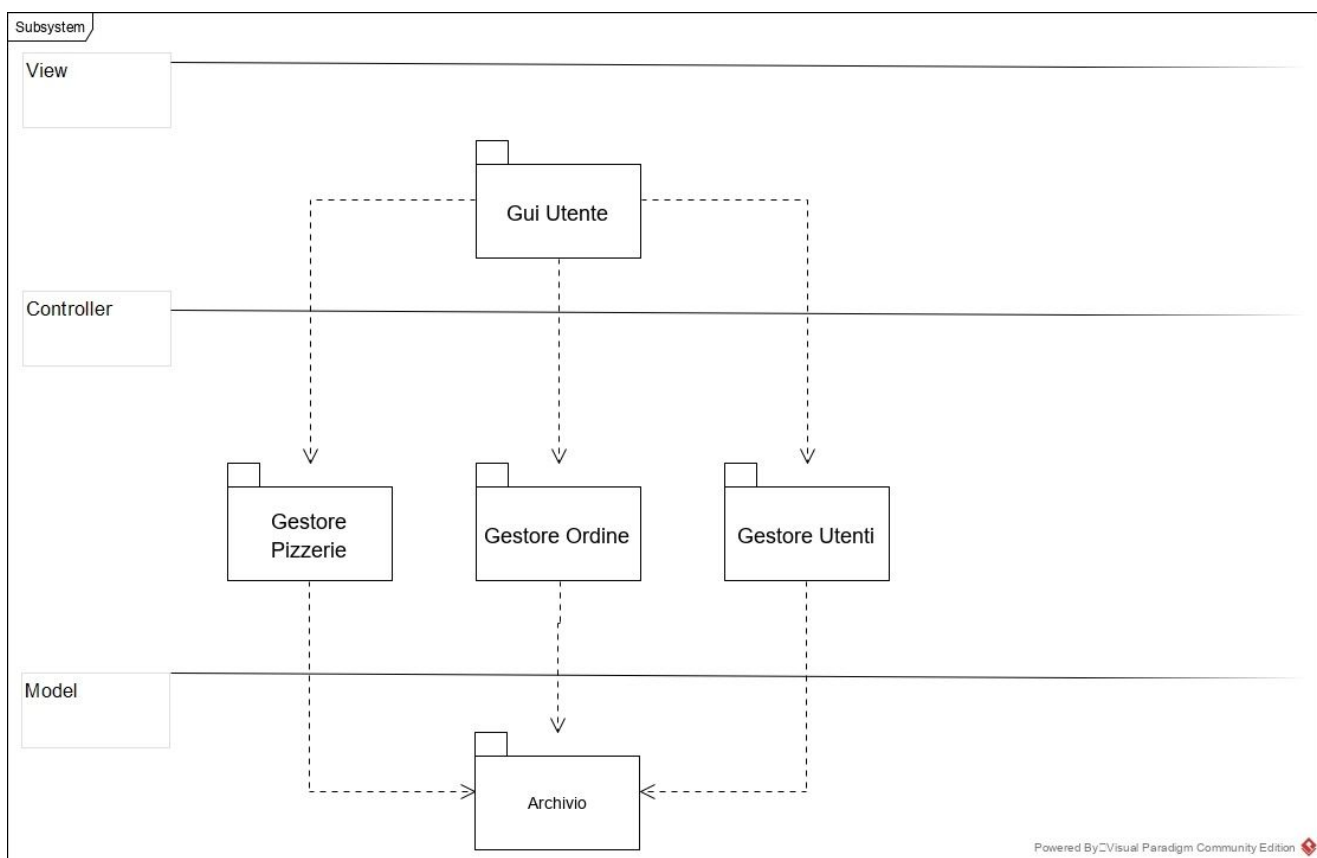
3.2.1. Decomposizione in Layer

La decomposizione prevista per il sistema è composta da tre layer che si occupano di gestirne aspetti e funzionalità differenti:

- View: raccoglie e gestisce elementi di interfaccia grafica e gli eventi generati su di essi;
- Controller: si occupa della gestione logica del sistema;
- Model: si occupa della gestione e dello scambio dei dati tra i sottosistemi;



3.2.2. Decomposizione in sottosistemi



3.3. Hardware/software mapping

Il sistema che si desidera sviluppare utilizzerà una struttura hardware costituita da un Server che risponderà ai servizi richiesti dai client.

Il client è una qualsiasi macchina dotata di web browser e connessione ad internet, che un utente può utilizzare per collegarsi alla piattaforma mentre il server è una macchina che gestisce la logica e i dati persistenti all'interno del database. Client e server utilizzeranno richieste HTTP per poter comunicare, il client inoltra richieste al server il quale provvederà a fornire i servizi richiesti.

Il server così come il client, dovranno avere una connessione internet, il server deve poter essere in grado di immagazzinare una quantità di dati non troppo elevata. Quindi è necessario un DBMS per poter gestire questi dati.

Per il modello mvc si è scelto questo tipo di configurazione:

1. Utilizzare per la parte di interfaccia, View, JSP, HTML, CSS, JavaScript e JQuery.
2. Per la parte di business logic, Control, utilizziamo Apache Tomcat e Java.
3. Per la parte di persistenza, e quindi di Data Access, utilizziamo un il DBMS MySql con JDBC.
4. Per la comunicazione si utilizza il protocollo HTTP, TCP/IP.

3.4. Persistent data management

Per la memorizzazione dei dati si è scelto un Database relazionale che consente un efficiente accesso ai dati, tempi di risposta brevi e capacità di memorizzazione elevate. Viene inoltre garantito l'accesso concorrente, quindi la sicurezza, in quanto solo gli utenti autorizzati possono accedervi.

E' stato deciso, quindi, di rendere persistente le seguenti Entità:

- Pizzeria
- Indirizzo
- Carta
- Ordine
- Prodotto
- Categoria
- Richieste Affiliazioni

- Recensione
- Cliente
- Ristoratore
- Utente

Pizzeria

Nome	Tipo	Null	Key
Nome	Varchar(20)	Not null	
Telefono	Varchar(15)	Not null	
Descrizione	Varchar(500)	Not null	
Indirizzo	int		Foreign Key
Proprietario	Varchar		Foreign Key
Carta	Varchar	Not null	Foreign Key
Partita Iva	Varchar(25)	not null	Primary Key
Orario apertura	time		
Orario chiusura	time		
Giorni apertura	varchar(50)		
Descrizione	varchar(45)		

Prodotto

Nome	Tipo	Null	Key
Nome	varchar(20)	Not Null	Primary Key
Prezzo	float	Not null	
Disponibilità	varchar(5)	Not null	
Ingredienti	varchar(50)	Not null	
Pizzeria	int	Not null	Foreign Key,

			Primary Key
Categoria	varchar(20)	Not null	Foreign Key

Indirizzo

Nome	Tipo	Null	Key
Via	varchar(40)	Not null	
Cap	varchar(5)	Not null	
Civico	varchar(6)	Not null	
Città	varchar(20)	Not null	
Utente	varchar(20)	Not null	Foreign Key
id	int	Not null, auto_increment	Primary Key

Carta

Nome	Tipo	Null	Key
Numero Carta	varchar(16)	Not null	Primary Key
Scadenza	varchar(5)	Not null	
Cvc	varchar(3)	Not null	
Intestatario	varchar(30)	Not null	
Utente	varchar(40)	Not null	Foreign Key

Ordine

Nome	Tipo	Null	Key
Id	int	Not null, auto_increment	Primary Key
Numero ordine	int	Not null	
Stato	varchar(30)	Not null	
Totale	float	Not null	
Tipo pagamento	int	Not null	

Data	timestamp	Not null	
Cliente	varchar(40)	Not null	Foreign Key
Indirizzo	int		Foreign Key
Tracker	varchar(5)		
Fattorino	varchar(30)		Foreign Key
Pizzeria	varchar(25)	Not null	Foreign Key
Tipo ordine	int	Not null	
Carta	varchar(40)		
Recensione	int		Foreign Key

Categoria

Nome	Tipo	Null	Key
Nome	varchar(20)	Not Null	Primary Key
Iva	int	Not null	

Richiesta Affiliazione

Nome	Tipo	Null	Key
Nome	varchar(30)	Not null	
Nome pizzeria	varchar(30)	Not null	
Commento	varchar(30)	Not null	
Data	date	Not null	
Stato	varchar(30)	Not null	
Email	varchar(30)	Not null	
Password	varchar(8)	Not null	
Cognome	varchar(30)	Not null	
Partita Iva	varchar(30)	Not null	

Telefono	varchar(30)	Not null	
Id Richiesta	int	Not null, auto_increment	Primary Key

Recensione

Nome	Tipo	Null	Key
Commento	varchar(500)	Not null	
Date	date	Not null	
Starring	int	Not null	
Id	int	Not null, auto_increment	Primary Key

Utente

Nome	Tipo	Null	Key
Email	varchar(30)	Not null	Primary Key
Password	varchar(25)	Not null	
Nome	varchar(20)	Not null	
Cognome	varchar(30)	Not Null	
Tipo	int	Not null	
Partita iva	varchar(11)		
Telefono	varchar(15)	Not null	
Riferimento pizzeria	varchar(25)		Foreign Key

3.5. Access control and security

Link Riferimento:

<https://docs.google.com/spreadsheets/d/1cqXGMjLScHEa02TbBoCT8OZJ3xfkE1oCl3Gc84xjgM/edit#gid=2035301975>

3.6. Global software control

Il controllo del flusso globale del sistema Speedy Pizza è di tipo event-driven, è necessaria una interazione dell'utente per poter utilizzare le funzionalità del sistema.

3.7. Boundary conditions

Startup Server

Per l'avvio del sistema Speedy Pizza è necessario che l'amministratore si occupi di avviare prima il Database Server, che si occupa della gestione dei dati persistenti e poi il Web Server.

Shutdown Server

È possibile terminare il sistema se e solo se tutti i sottosistemi sono stati disattivati in precedenza. Prima della disattivazione totale del database-server e dell'application-server verranno disconnessi tutti i client connessi al sistema; ad ognuno di loro arriverà una notifica prima della terminazione del sistema. Quindi lo stato dei client non verrà salvato.

Lo shutdown può essere effettuato nel caso in cui si debba effettuare manutenzione del sistema.

Exception

- Un malfunzionamento del sistema può essere dovuto ad un'interruzione della connessione tra Web Server e Database Server o anche tra client e Web Server.
- Inoltre un malfunzionamento può essere dovuto anche ad errori in fase di implementazione.
- Un altro caso di fallimento potrebbe essere dovuto ad un errore critico nell'hardware, come per esempio l'arresto improvviso di uno o entrambi i Server, in questo caso non è prevista alcuna contromisura.

Nel caso in cui si dovesse verificare un'interruzione di connessione tra Web Server e Database Server, allora si ritenta la connessione fino a che non viene ristabilita. Se così facendo il problema non dovesse essere risolto allora sarà compito dell'Amministratore eseguire le giuste procedure.

Nel secondo caso, invece, l'Amministratore effettuerà le giuste verifiche dell'errore e provvederà a risolverlo, se possibile.

Nel terzo caso, invece, è previsto un check di consistenza dei dati all'interno del DB. Nel caso in cui ci siano degli errori, allora verrà effettuato un ripristino dei dati danneggiati.

ID:	UC_01
Use Case Name:	Startup Server
Participating actor:	Amministratore
Entry condition:	<ul style="list-style-type: none"> • L'amministratore ha effettuato l'accesso ai Server
Flow events:	<ol style="list-style-type: none"> 1. L'amministratore avvia il servizio di Esecuzione del Database Server 2. Se il Database server era stato spento correttamente allora viene avviato il servizio di esecuzione. Nel caso in cui il Database Server era stato spento in modo anomalo, viene eseguito il check di integrità sui dati, quindi vengono ripristinati i dati corrotti. 3. L'amministratore avvia il servizio di esecuzione del Web Server 4. Il Web server si avvia correttamente
Exit condition:	<ul style="list-style-type: none"> • I server vengono avviati correttamente e restano in attesa di connessioni
Extension point	

ID:	UC_02
-----	-------

Use Case Name:	Shutdown Server
Participating actor:	Utente
Entry condition:	<ul style="list-style-type: none"> • L'Utente si trova sulla home della piattaforma
Flow events:	<ol style="list-style-type: none"> 1. L'Utente clicca su "Ordina" 2. Il sistema mostra la pagina dell'ordine 3. L'utente visualizza correttamente la pagina dell'ordine ,inserisce l'indirizzo nella barra di ricerca e clicca su "Cerca" 4. Il sistema procede alla verifica dell'indirizzo e alla successiva validazione utilizzando un servizio esterno, quindi mostra tutte le pizzerie vicine all'indirizzo ricercato e la mappa con la posizione dell'indirizzo ricercato
Exit condition:	<ul style="list-style-type: none"> • L'Utente visualizza correttamente la lista delle pizzerie
Extension point	Nel caso in cui al punto 4, la verifica dell'indirizzo non vada a buon fine rifarsi al caso d'uso UC_25

4. Subsystem services

Sottosistema Gestore Utenti	
Servizi Offerti a Gestore Ordini	
addProductToCart	Servizio che consente l'aggiunta di un prodotto al carrello
deleteProductFromCart	Servizio che consente la rimozione di un prodotto dal carrello
showOrders	Servizio che consente di prelevare gli ordini effettuati, sia del cliente, sia del ristorante che del fattorino
assegnaFattorino	Servizio che consente al gestore utenti di assegnare un fattorino ad una pizzeria
checkout	Servizio che consente di effettuare il checkout dell'ordine
showCart	Servizio che consente di visualizzare il carrello
selectCardOrder	Servizio che consente di scegliere la carta per effettuare il checkout
selectAddressOrder	Servizio che consente di scegliere un indirizzo di consegna per l'ordine
modificaStatoOrdine	Servizio che consente di modificare lo stato dell'ordine

Sottosistema Gui Utente	
Servizi Offerti a Gestore Utenti	
Login	Servizio che consente il login degli utenti
Logout	Servizio che consente il logout degli utenti
showHome	Servizio che consente di visualizzare la home dell'utente
showProfile	Servizio che consente di visualizzare il profilo dell'utente
eliminaUtente	Servizio che consente l'eliminazione di un utente
showUtenti	Servizio che consente la visualizzazione degli utenti
inserisciFattorino	Servizio che consente l'inserimento di un fattorino
insertCard	Servizio che consente l'inserimento di una carta di credito
insertAddress	Servizio che consente l'inserimento di un indirizzo di consegna
modificaProfilo	Servizio che consente la modifica dei dati del profilo
showRichiesteAff.	Servizio che consente la visualizzazione delle richieste di affiliazione
acceptRichieste	Servizio che consente di accettare una richiesta di affiliazione

declineRichieste	Servizio che consente di rifiutare una richiesta di affiliazione
------------------	--

Sottosistema Gui Utente	
Servizi Offerti a Gestore Pizzerie	
showCatalogo	Servizio che consente di visualizzare il catalogo di una pizzeria
searchPizzerie	Servizio che consente di ricercare pizzerie
addProductToCatalogo	Servizio che consente di aggiungere un prodotto al catalogo
modificaCatalogo	Servizio che consente la modifica del catalogo
deleteProduct	Servizio che consente di eliminare un prodotto dal catalogo
modificaProdotto	Servizio che consente di modificare un prodotto
modificaRicezioneOrdini	Servizio che consente di modificare la ricezione degli ordini