Progetto
Ingegneria del Software

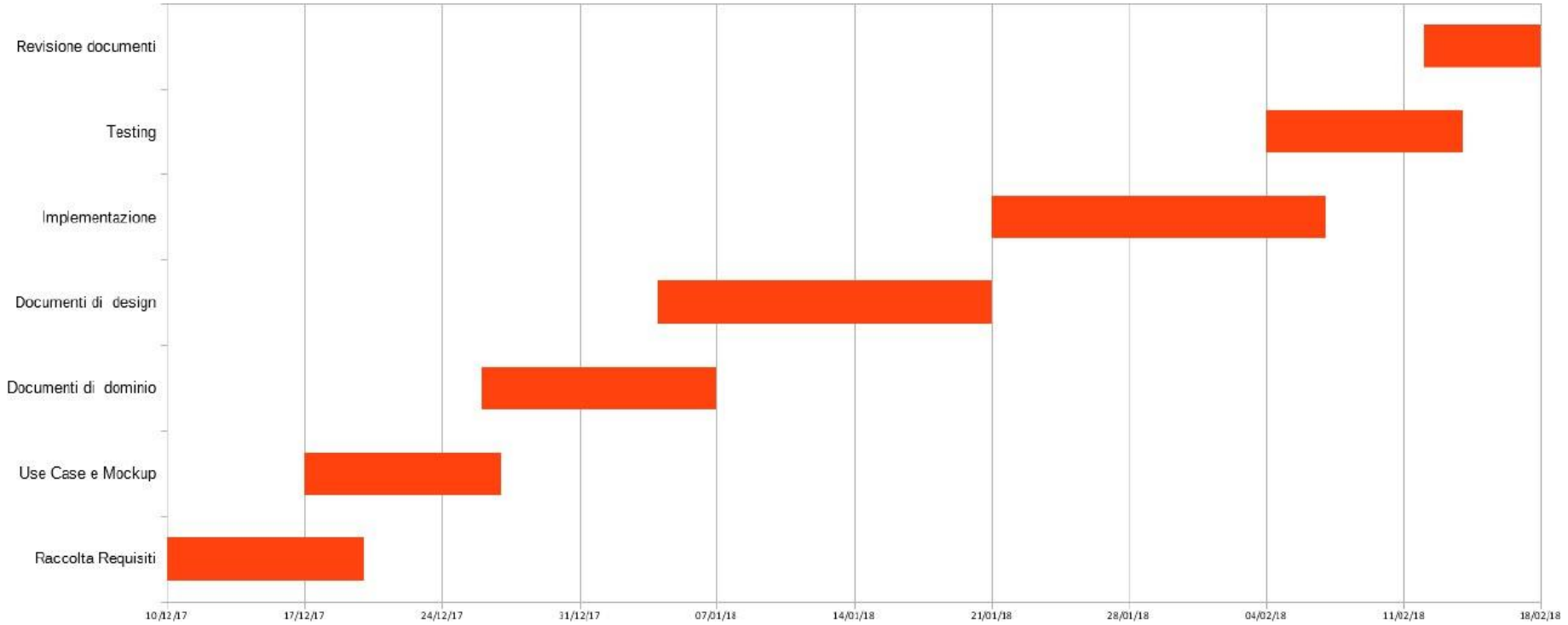
gruppo 17

Stefano Falangone	N86002437
Alessandro Caputo	N86001934
Crescenzo Cipolletta	N86001866

L’obiettivo

- Realizzare un software per gestire clienti, eventi e luoghi con grande partecipazione di pubblico
- Fornire statistiche sull’andamento complessivo del business e informazioni su singoli eventi, clienti, luoghi

Progettazione di Gantt



Requisiti funzionali

- Operazioni di CRUD per Clienti, Eventi, Luoghi
- Visualizzazione statistiche singole
- Visualizzazione statistiche generali
- Interpolazione dei prezzi degli eventi

Vincoli di eliminazione

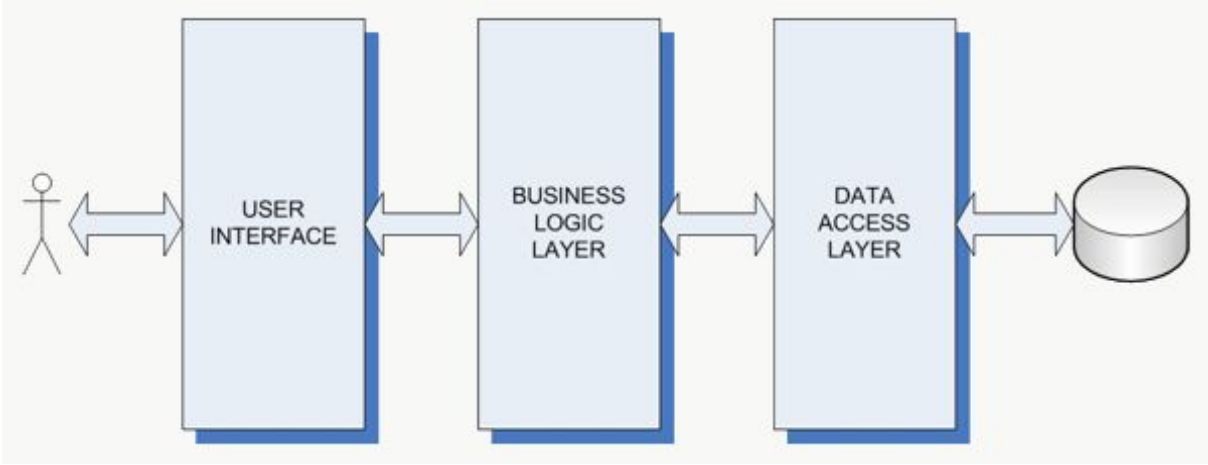
- Un evento è eliminabile solo se non vi sono biglietti venduti
- Un luogo è eliminabile solo se non ospita eventi della categoria sovraesposta
- Eliminare un cliente deve provocare la cancellazione dei biglietti associati

Requisiti non funzionali

- Ogni funzionalità accessibile in 3 click dalla schermata iniziale
- Il software si deve interfacciare con DynamoDB di AWS
- Software utilizzabile da più utenti in contemporanea
- Software compatibile con Windows, macOS e Linux

Architettura del sistema

Adottata architettura client/server a tre livelli



Architettura interna: ECB

Le classi sono state suddivise in Entity - Boundary - Control

- Entity: oggetti da manipolare con un corrispettivo “reale”
- Boundary: interfacce e metodi di interazione tra Utente e software
- Control: gestori della logica con cui si modellano le entità

Design Pattern: DAO

Per disaccoppiare la business logic dal database si è utilizzato il Data Access Object. Si è introdotto un layer tra controller e database per il quale ogni richiesta di un controller per il database viene inviata al DAO, il quale ritorna delle entity che sono indipendenti dal database utilizzato

Vantaggi del DAO: progettato per il cambiamento

Qualora il committente desideri cambiare database (ad esempio scegliendone uno relazionale) è possibile modificare soltanto le classi DAO andando quindi a riscrivere soltanto una frazione del codice complessivo

Alta coesione e basso accoppiamento...

Obiettivo costante è stato mantenere le classi quanto più coese e quanto meno accoppiate possibili

Soluzione: progettazione guidata dalle responsabilità delle singole classi e legge di Demetra per disaccoppiare quanto possibile

...con qualche limite

Il progetto si è rivelato essere ricco di entity e controller, è stato spesso necessario far comunicare i controller tra di loro

Esempi:

- Evento eliminabile solo se non vi sono biglietti venduti (quindi eventoController deve chiedere a bigliettoController)

...con qualche limite

- Un luogo è eliminabile solo se non vi sono ospitati eventi con biglietti venduti
- Una statistica è ottenuta “incrociando” dati diversi (un luogo quanti eventi ha ospitato, oppure quanti biglietti complessivamente sono stati venduti per eventi in quel luogo)

Database: DynamoDB

Database offerto da Amazon Web Services nella tier gratuita (S3) di tipo NoSQL

DynamoDB

amazon web services™

Database: approccio noSQL

- Mancanza di qualunque vincolo del database (no Foreign Key, no chiavi Univoche, no “on delete cascade”)
- Ogni vincolo responsabilità del developer
- Approccio column-oriented
- Unico vincolo presente: primary key su un attributo

Package e libraries

- Le classi sono state suddivise in package rispettando la suddivisione ECB (con un package aggiuntivo di testing e uno di connessione)
- Sono state utilizzate due librerie free esterne: jFreeChart e LGoodDatePicker. La prima per la visualizzazione di statistiche e la seconda per introdurre l'uso di un calendar durante la selezione di una data.

Piano di Testing

1. Verifica delle funzionalità offerte (CRC cards)
2. Testing attraverso jUnit

CRC cards

Una CRC card per ogni Use-Case definito con template di Cockburn che copra tutte le extensions e le subvariations

Testing attraverso jUnit

Testing di ogni metodo dei DAO ad eccezione delle eliminazioni.

Problema: l'eliminazione non fa mantenere l'integrità del database

Testing attraverso jUnit

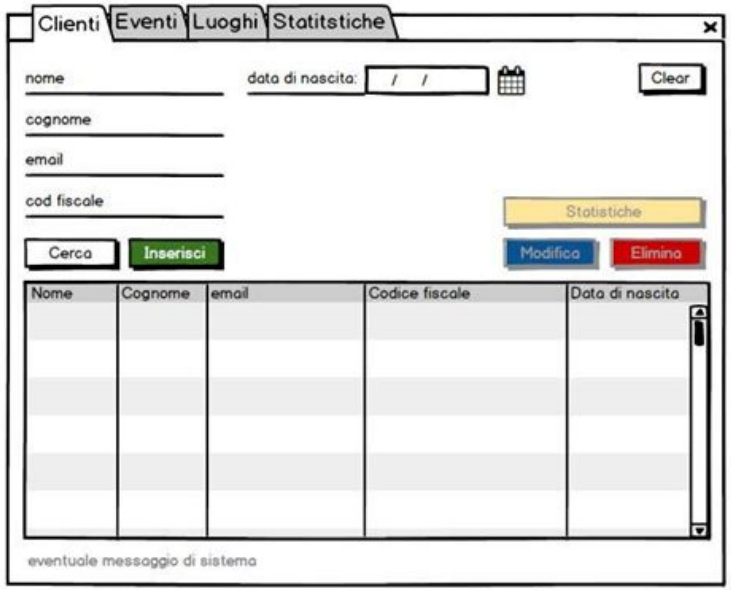
Testing di ogni metodo dei DAO ad eccezione delle eliminazioni.

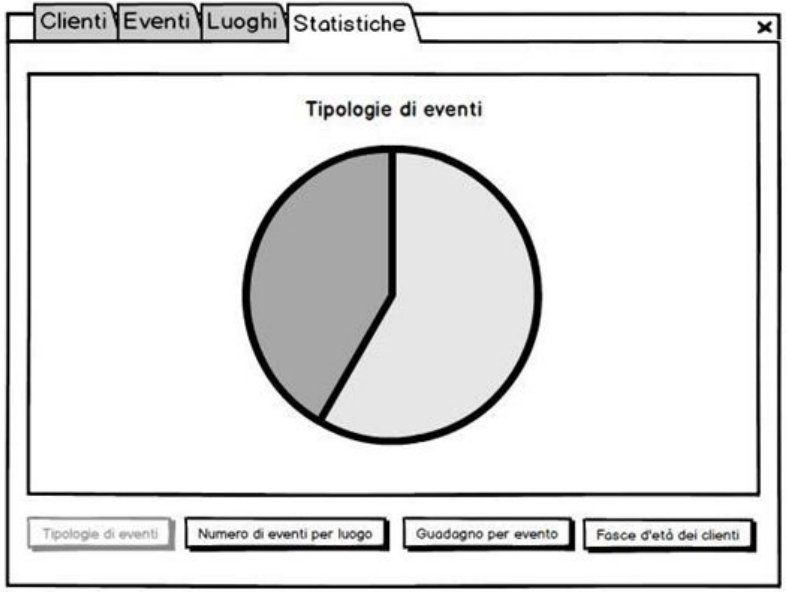
Problema: l'eliminazione non fa mantenere l'integrità del database

Soluzione ideale: sviluppare una Test Suite che svolge inserimenti prima di eseguire il testing, dopodiché al termine elimina ciò che non c'era inizialmente in DynamoDB

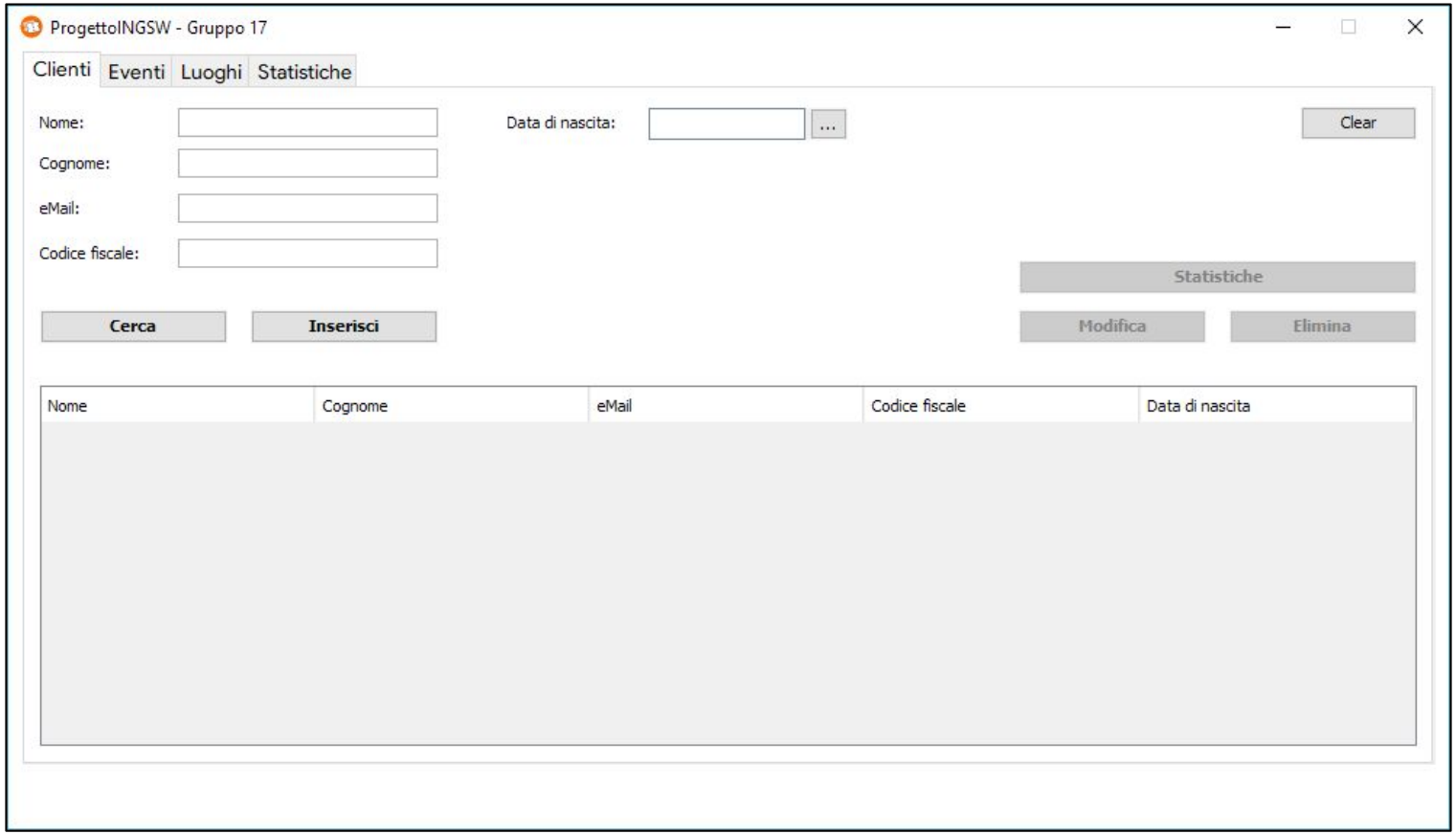
Idea di interfaccia

mock-up

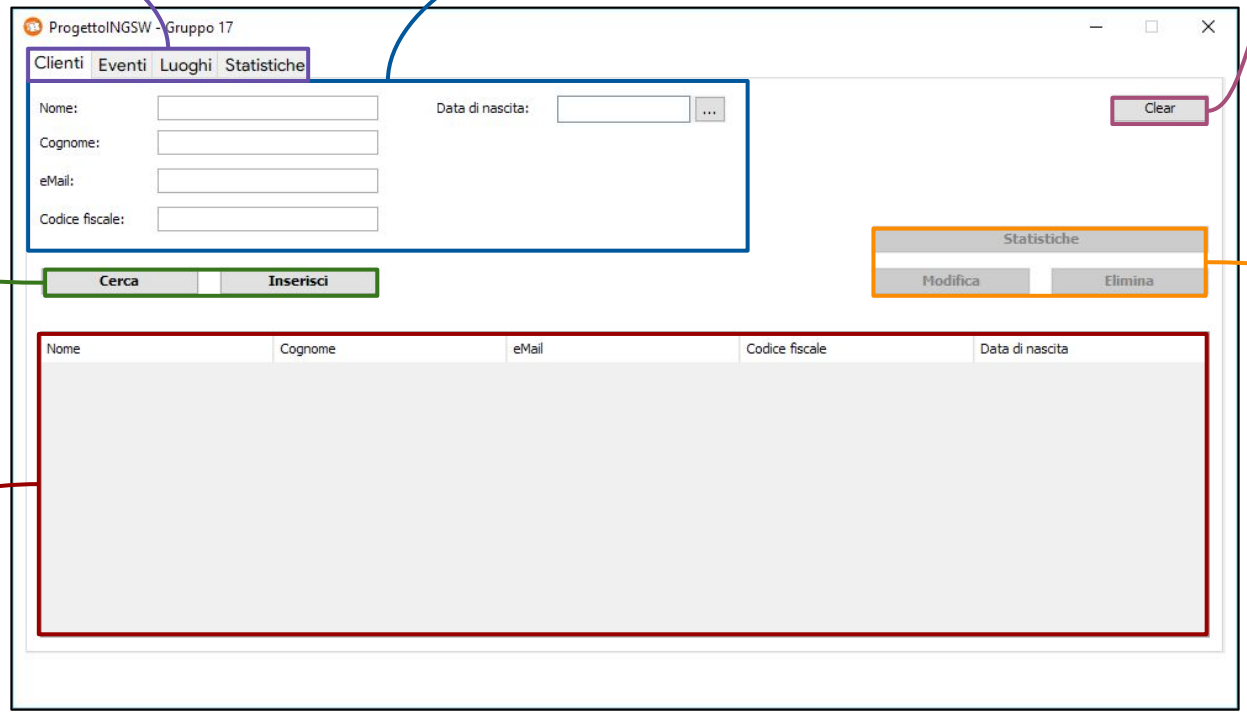




Interfaccia utente



Interfaccia utente



CATEGORIE

CAMPI DI INPUT

PULISCE I CAMPI E LA TABELLA DEI RISULTATI

PULSANTI

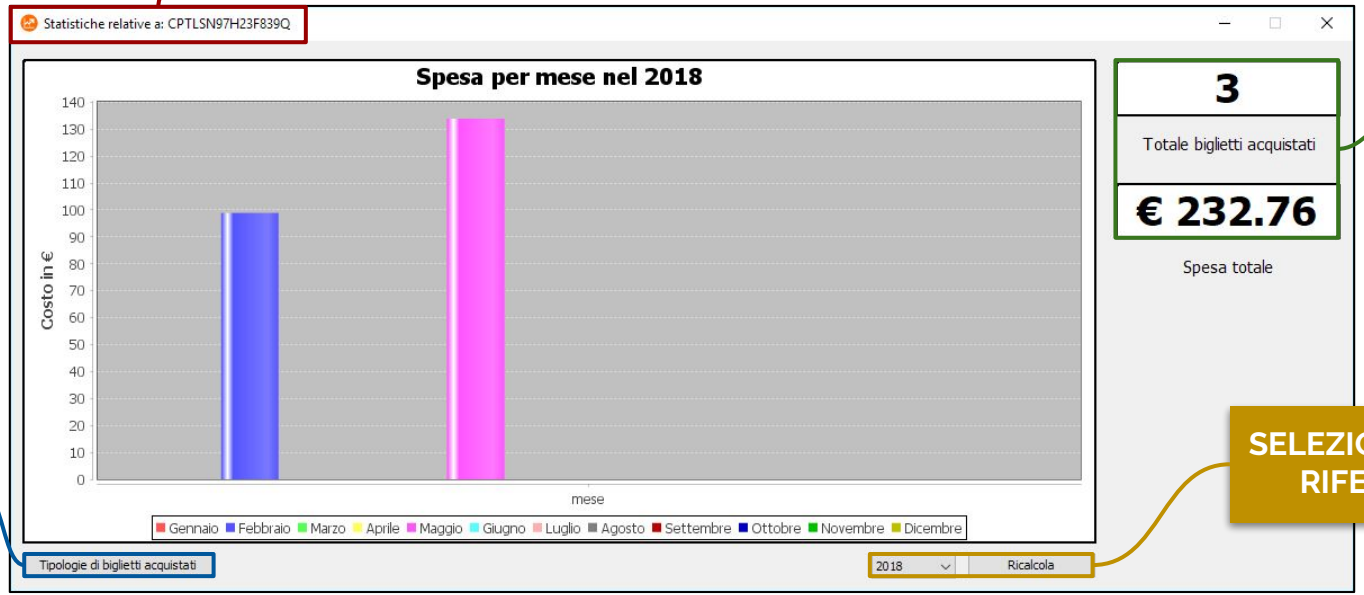
PULSANTI ABILITATI QUANDO SI SELEZIONA UN ELEMENTO DALLA TABELLA DEI RISULTATI

RISULTATI DELLA RICERCA

Statistiche Cliente: spesa per mese

CODICE FISCALE DEL CLIENTE SELEZIONATO

INFORMAZIONI COMPLESSIVE



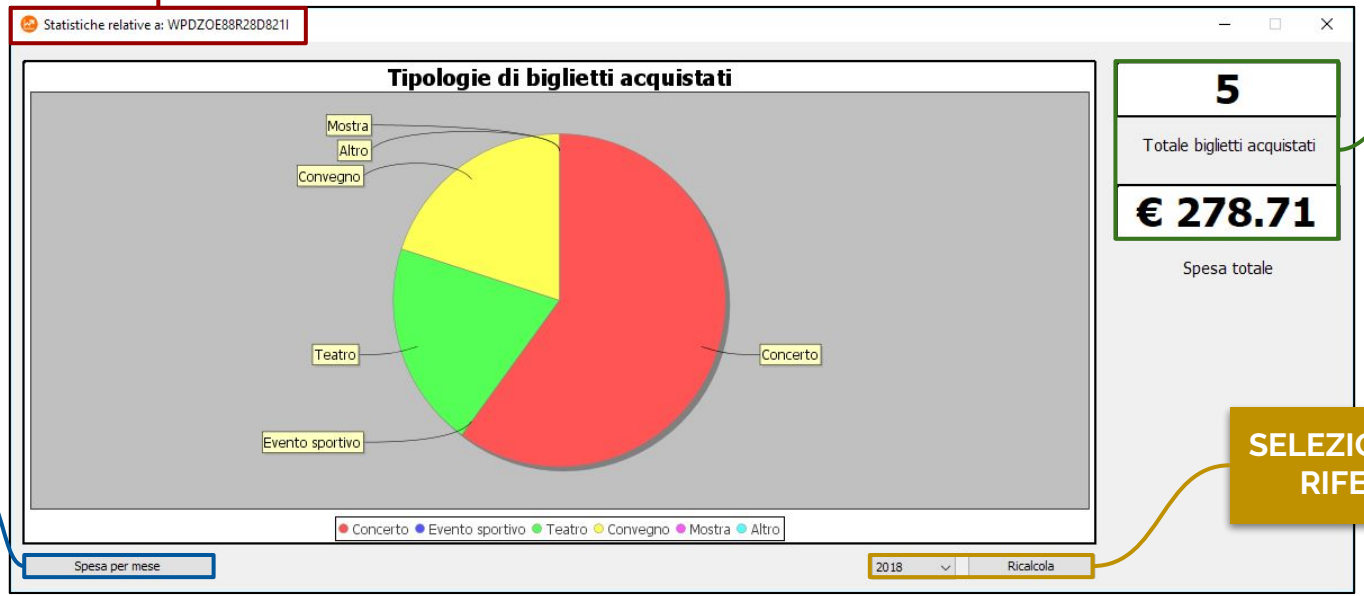
VISUALIZZA SECONDO GRAFICO

SELEZIONE ANNO DI RIFERIMENTO

Statistiche Cliente: tipologie di biglietti acquistati

CODICE FISCALE DEL CLIENTE SELEZIONATO

INFORMAZIONI COMPLESSIVE



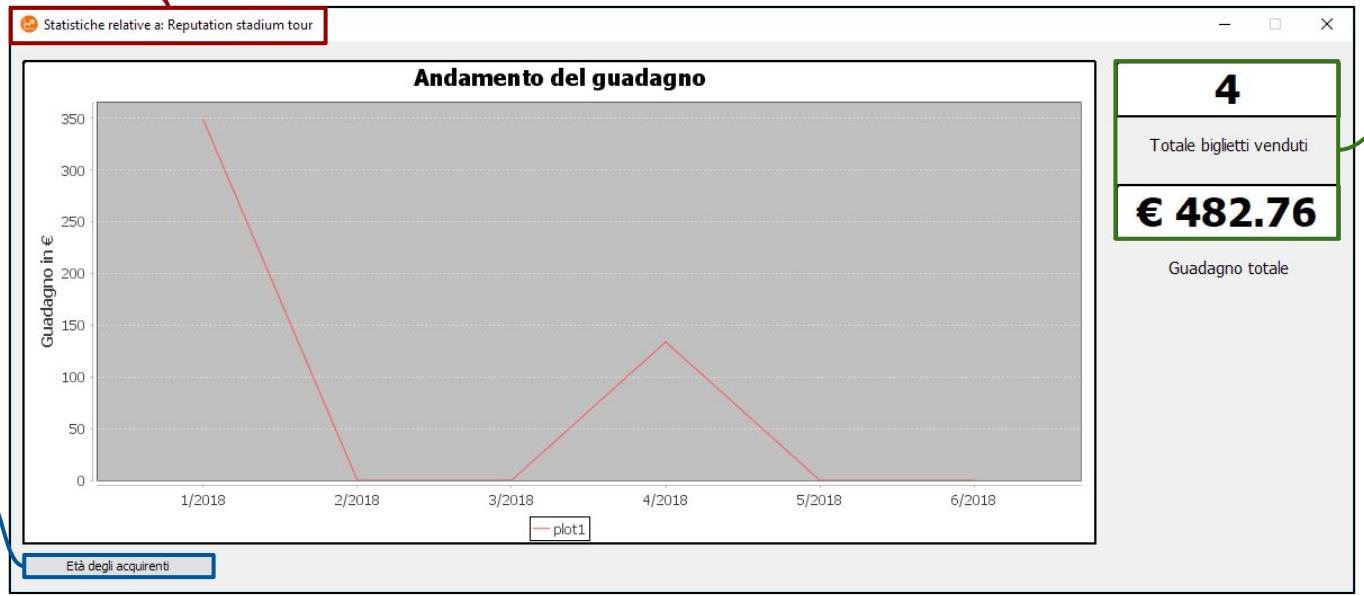
VISUALIZZA PRIMO GRAFICO

SELEZIONE ANNO DI RIFERIMENTO

Statistiche Evento: andamento del guadagno

EVENTO SELEZIONATO

INFORMAZIONI COMPLESSIVE

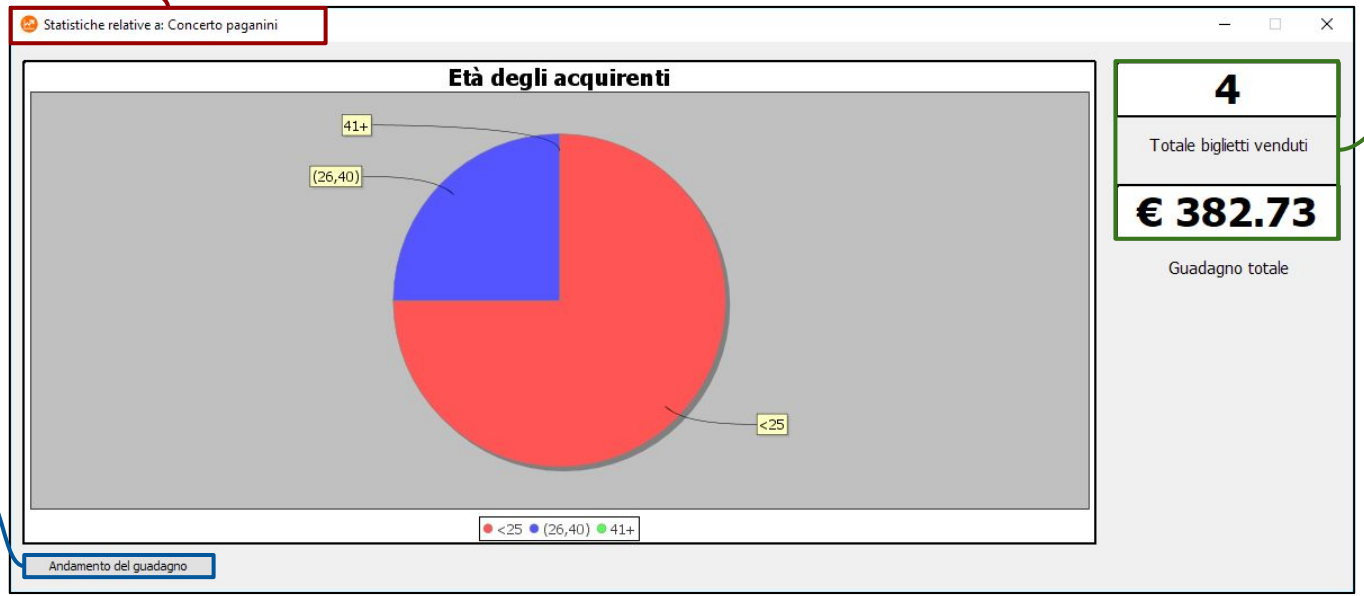


VISUALIZZA SECONDO GRAFICO

Statistiche Evento: età degli acquirenti

EVENTO SELEZIONATO

INFORMAZIONI COMPLESSIVE

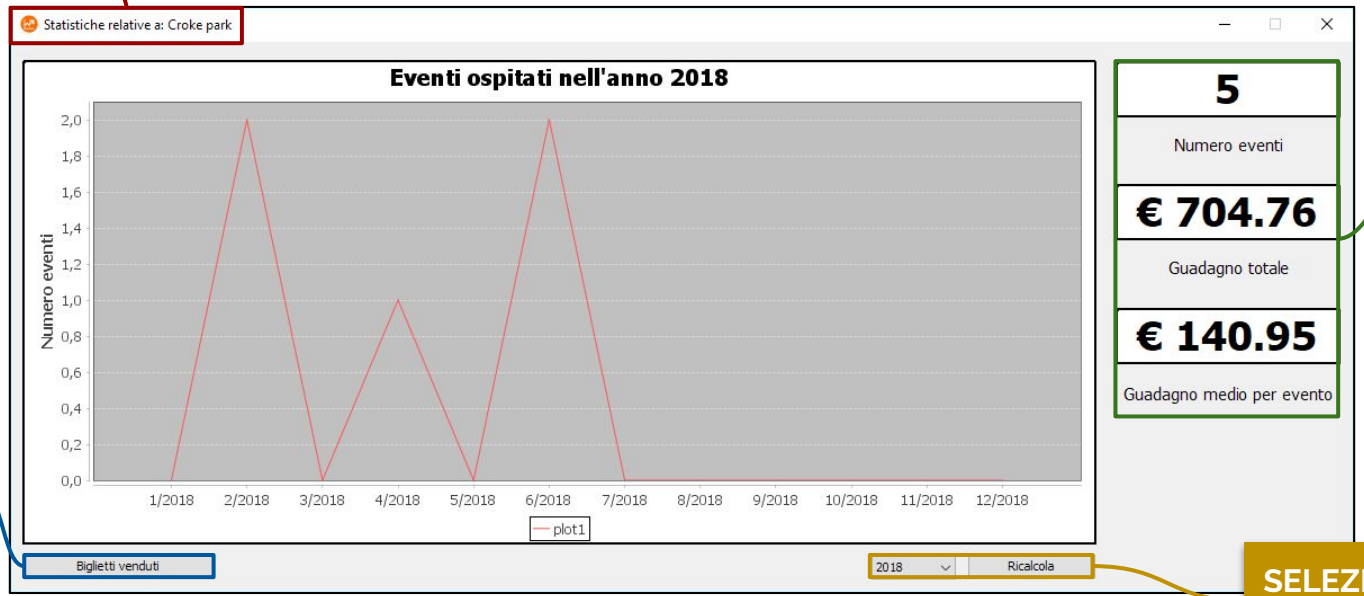


VISUALIZZA PRIMO GRAFICO

Statistiche Luogo: eventi ospitati per anno

LUOGO SELEZIONATO

INFORMAZIONI COMPLESSIVE



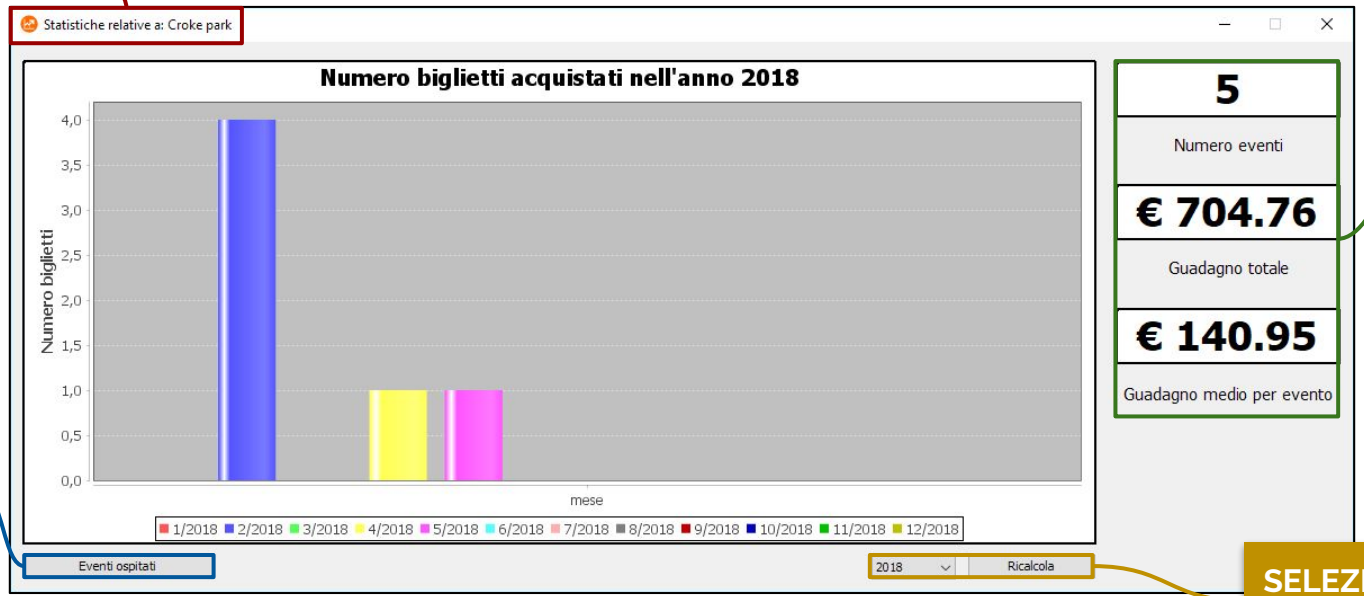
VISUALIZZA SECONDO GRAFICO

SELEZIONE ANNO DI RIFERIMENTO

Statistiche Luogo: biglietti acquistati per anno

LUOGO SELEZIONATO

INFORMAZIONI COMPLESSIVE



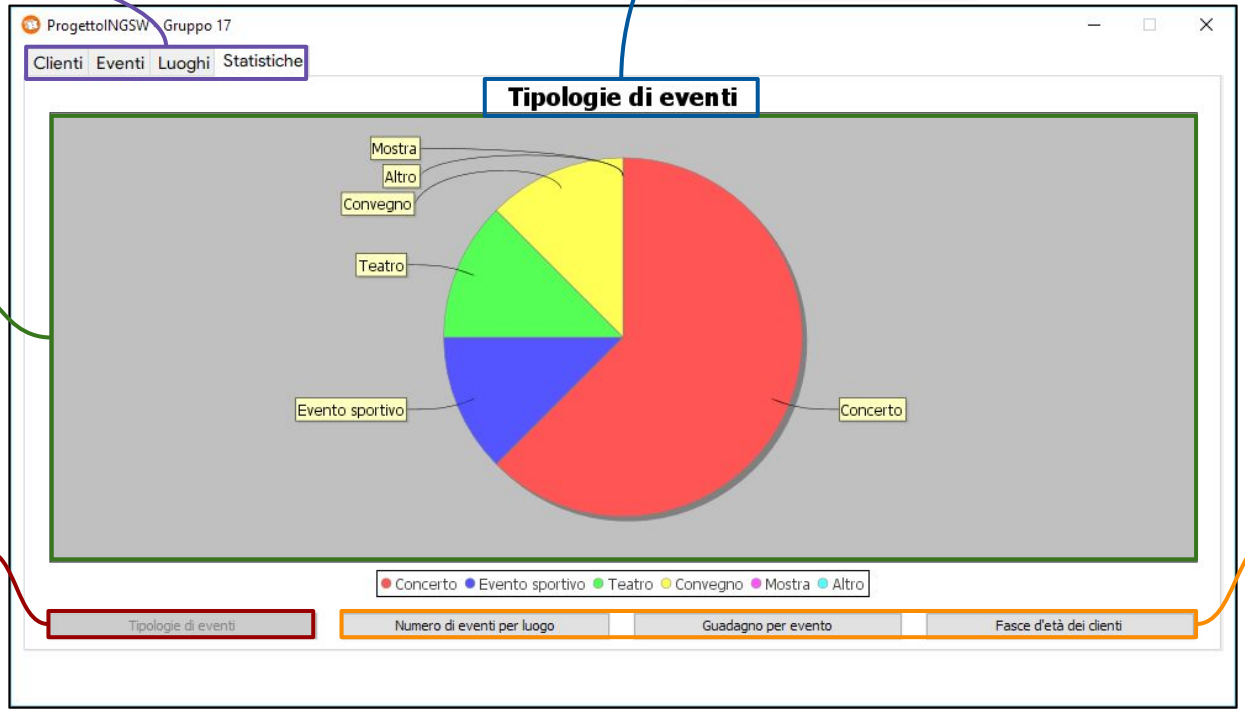
VISUALIZZA PRIMO GRAFICO

SELEZIONE ANNO DI RIFERIMENTO

Statistiche generali: Tipologie di eventi

CATEGORIE

STATISTICA CORRENTE

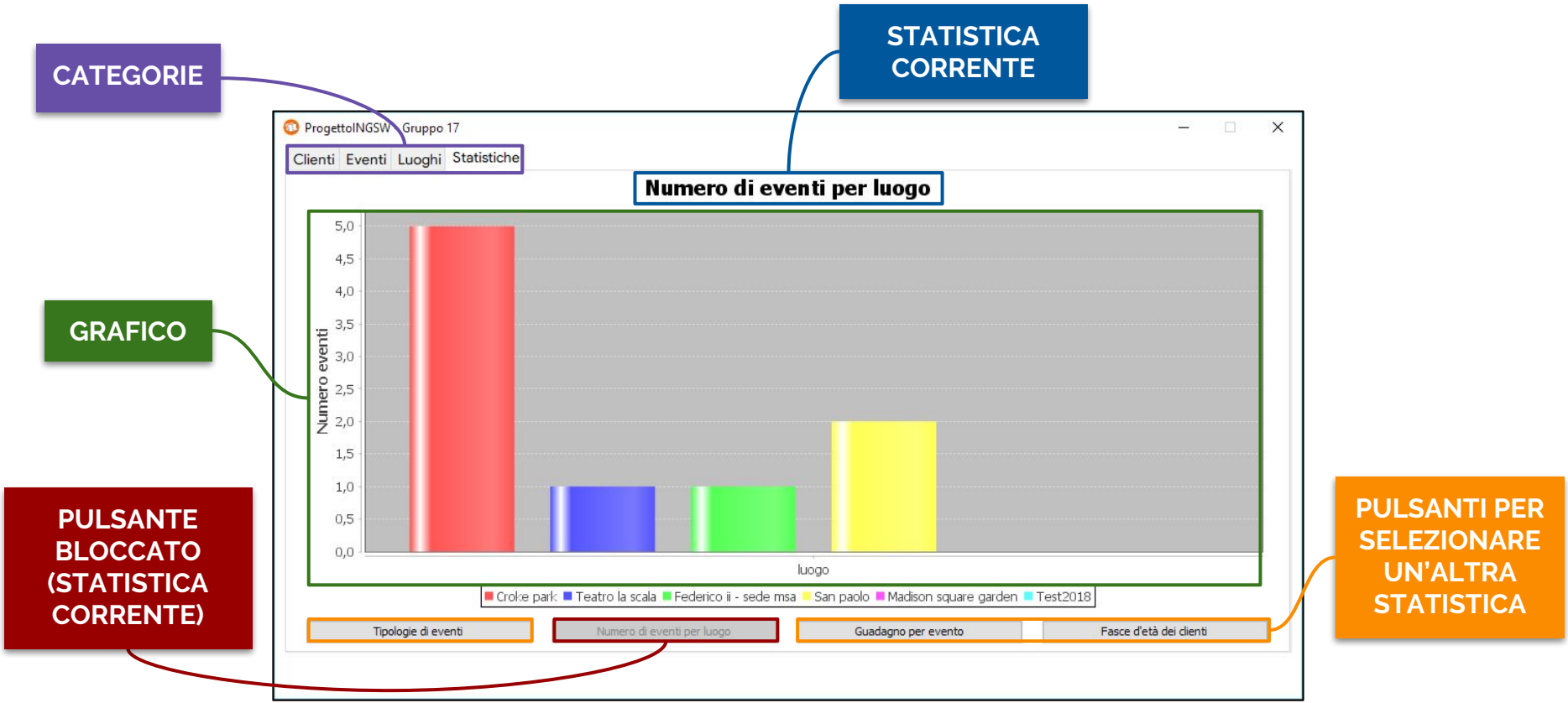


GRAFICO

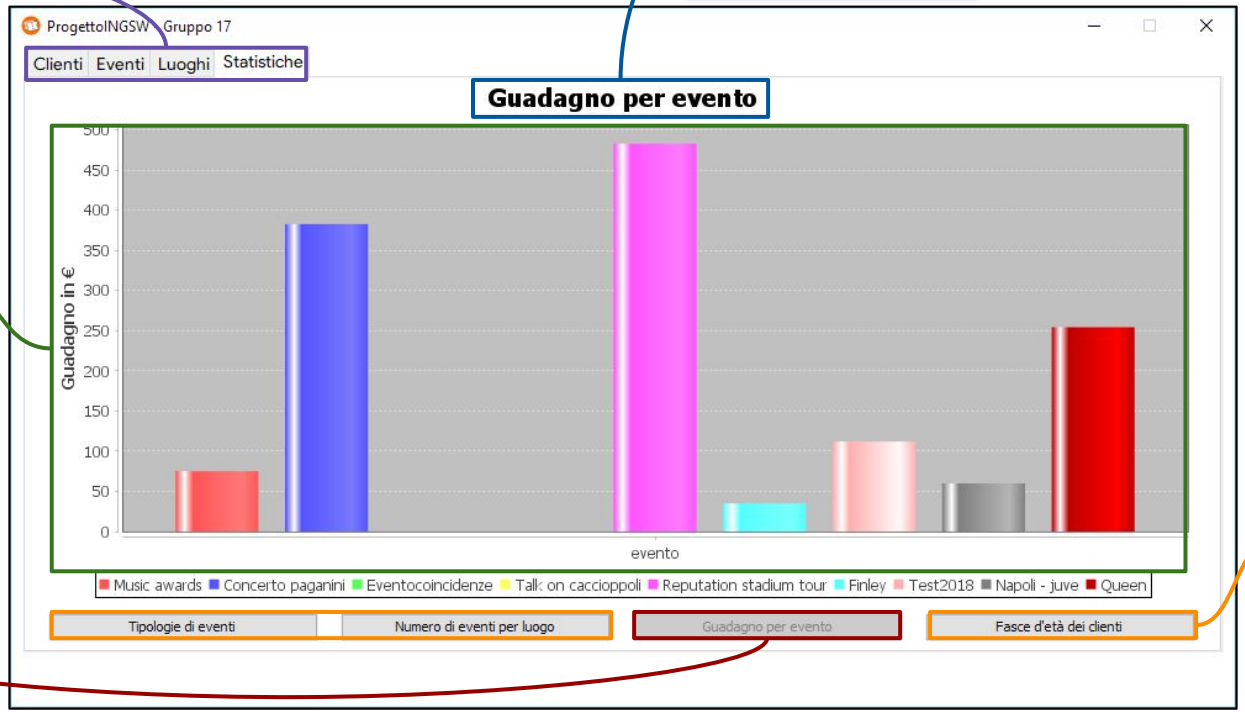
PULSANTE BLOCCATO (STATISTICA CORRENTE)

PULSANTI PER SELEZIONARE UN'ALTRA STATISTICA

Statistiche generali: Numero di eventi per luogo



Statistiche generali: Guadagno per evento



Statistiche generali: Fasce d'età dei clienti