UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base

Corso di Laurea in Ingegneria Informatica

Elaborato finale in **Basi di Dati**

Oracle APEX: Esempio d'uso

Anno Accademico 2018-2019

Candidato:
Alfredo Fiore
Matr. N46002791

A chi, in un modo o nell'altro, c'è sempre stato…

"You can have data without information,
but you cannot have information without data."

- Daniel Key

Indice

Oracle APEX: Esempio d'uso 1
Introduzione 1
Capitolo 1: Introduzione alle applicazioni Web 3
1.1 DBMS 4
1.2 DBMS Web Based 5
1.3 Framework Applicati alle Web App 5
Capitolo 2 : Oracle APEX 8
2.1 Gestione dei dati e creazione degli oggetti 9
2.1.1 Creazione di oggetti 9
2.1.2 Creazione di una tabella 10
2.1.3 Creazione di un Trigger 12
2.1.4 Creazione di una vista materializzata 14
2.2 Creazione dell'applicazione 15
2.2.1 Inizializzazione dell'app 15
2.2.2 Aggiunta di pagine 17
Capitolo 3 : Esempio di applicazione 23
3.1 Scelte progettuali 23
3.2 Avvio dell'applicazione 24
3.3 Funzionalità implementate 25
3.3.1 Grafici relativi ai medici e alle prenotazioni 26
3.3.2 Lista Prenotazioni e relativo form di inserimento 27
3.3.3 Descrizione Generale delle prenotazioni 29
3.2.4 Calendario Appuntamenti 30
Conclusioni 31
Bibliografia 36
Appendice 32
A1 Creazione delle tabelle e definizione dei vincoli di integrità referenziale 32
A2 Popolamento delle tabelle 33
A2.1 Tabella Pazienti 33
A2.2 Tabella Medici 34
A2.3 Tabella Prenotazioni 34
A3 Queries omesse nella trattazione 35

Introduzione

L’obiettivo di questa tesi è quello di analizzare il tool per lo sviluppo di applicazioni Web: Oracle APEX[1].

Il tool APEX permette il design lo sviluppo di applicazioni basate su databases(sia locali che Cloud), utilizzando una semplice interfaccia web.

In questo elaborato verranno esposte le funzionalità offerte da APEX, e verrà proposto un esempio di applicazione web basata su un database locale.

[1] APplication EXpress

Capitolo 1: Introduzione alle applicazioni Web

Con il termine applicazione web si intende un software con infrastruttura simile a quella di un sito web che consente agli utenti di interagire con esso. Generalmente si tratta di una applicazione basata sul paradigma client-server[2], ed offre svariati servizi all'utente client.

Le Basi di Dati forniscono un apporto chiave nello sviluppo di un' applicazione web, in quanto permettono la persistenza delle informazioni. La tipica architettura software di una web application è quella "a tre strati":

1) Presentation layer

Rappresenta il "terminale dell'applicazione", la parte visibile all'utente tramite cui può interagire col sistema. Comprende le componenti che si occupano di presentare l'informazione ai client utente, e che consentono agli stessi di interagire con il sistema per sottomettere operazioni ed ottenere risultati.

Le pagine sono solitamente realizzate con linguaggi come HTML, Javascript, JSP, etc., ed elaborate da un browser web.

2) Application Layer

Rappresenta la parte invisibile all'utente che si occupa della logica elaborativa del sistema.

Questo strato è sviluppato con linguaggi come Java, PHP, Javascript, etc.

Il livello applicazione controlla la funzionalità di un'applicazione eseguendo elaborazioni dettagliate.

[2] Architettura software basata su un componente Server che gestisce ed elabora richieste effettuate dai vari componenti Client.

3) Data layer

Rappresenta il database (MySQL, Oracle, …), per la persistenza delle informazioni. Il nome Data Layer viene utilizzato nello specifico quanto si utilizza un DBMS per implementare il livello.

1.1 DBMS

Un Database Management System, o DBMS è un sistema software progettato per consentire la creazione, l'interrogazione e la manipolazione di un database. Si tratta sostanzialmente di uno strato software posto tra l'utente ed i dati veri e propri. In questo modo, l'utente e le applicazioni non accedono alla rappresentazione fisica dei dati, ma ne vedono solamente una rappresentazione logica.

Un DBMS è uno strumento per la creazione e la gestione efficiente di grandi quantità di dati che consente di conservarli in modo sicuro per lunghi periodi di tempo.

I servizi forniti da un DBMS agli utenti sono:

1) **Persistent storage**: Un DBMS permette la memorizzazione di dati, anche in una quantità elevata, garantendo un'alta flessibilità.
2) **Programming interface**: Un DBMS permette l'accesso e la gestione dei dati attraverso un potente linguaggio di interrogazione.
3) **Transaction management**: UN DBMS permette l'accesso concorrente ai dati, in modo da evitare problemi all'interno dell'applicazione.

1.2 DBMS Web Based

Una delle principali applicazioni del DBMS è rappresentata dai DBMS Web Based, che vengono utilizzati per garantire la persistenza dei dati di siti o di Applicazioni online, che necessitano la gestione di numerose informazioni, come ad esempio siti di E-commerce.

Utilizzare un DBMS Web Based porta numerosi vantaggi, tra i quali:

- **Semplicità di utilizzo**: In molte applicazioni, come Oracle APEX, non è necessario avere un alto livello di conoscenza di linguaggi di programmazione.
- **Supporto Cross-Platform**: La persistenza dei dati è relativa al DBMS stesso, e non sussistono problemi legati alla compatibilità.
- **Scalabilità**: Possiamo facilmente modificare o aggiungere elementi al nostro sistema senza intaccarne la struttura di base.

1.3 Framework Applicati alle Web App

Con framework si intende la struttura di un programma che viene utilizzata come base nello sviluppo software. È una soluzione utilizzata dagli sviluppatori per evitare di ricominciare ogni volta da zero.
Per molte funzioni standard, infatti, esistono soluzioni già pronte. Un framework rappresenta una collezione di diverse classi collegate tra loro e definisce una struttura di base per il design di ogni software, che viene sviluppata a partire dal framework.

Un framework utilizzato come base per la realizzazioni di Web App, viene definito framework per applicazioni web.

I framework mettono a disposizione strutture chiare, che consentono a chi sviluppa l'applicazione web, di soddisfarne in modo semplice e rapido ogni requisito.
I principi progettuali su cui si basano i Framework sono i seguenti:

1) **Don't repeat yourself**: Si mira ad evitare la ridondanza di informazioni, quali ad esempio duplicati di codici, in quanto potrebbero riflettere negativamente sulla gestione software.

2) **Keep it short and simple**: Il principio si basa sull'idea che ad ogni problema corrisponda una soluzione, e nel caso in cui per un determinato problema, si trovino più percorsi risolutivi percorribili, si preferisce scegliere quella che ha il minor numero di ipotesi e variabili, per rendere il sistema semplice.

3) **Convention over configuration**: I framework dovrebbero stabilire un metodo grazie alle impostazioni standard, oltre ad offrire a chi lo usa altre possibilità di configurazione opzionali.

I Framework per applicazioni web risolvono solo i problemi relativi alla realizzazione delle interfacce, in quanto pongono delle basi per la creazione delle interfacce utente, necessarie alla visualizzazione, manipolazione e gestione dei dati.
Esempi di framework per applicazioni web sono, tra gli altri, Apache Struts e Swing.

Esistono altri tipi di Framework, come Oracle APEX, la cui funzione principale è quella di creare un'applicazione web, ma che permette anche di gestire dei dati, sia a livello locale, che a livello cloud.

Capitolo 2 : Oracle APEX

Oracle Application Express(APEX) è un tool di sviluppo utilizzato per la creazione di Applicazioni Web, minimizzando la complessità di progettazione. Questo strumento presenta infatti una serie di features che permettono una facile risoluzione di problemi comuni, pur non essendo programmatori esperti.

La sua principale funzionalità è quella di creare una semplice interfaccia grafica, con il quale un utente può interagire ed eseguire operazioni. APEX permette inoltre allo sviluppatore di popolare il database al quale l'applicazione si riferirà, sia conservando in locale i dati, sia attingendo a un database cloud senza la necessità di sviluppare codice eseguibile. All'interno del *Sql Workshop*, sezione apposita dell'applicazione, lo sviluppatore può facilmente creare tabelle, triggers e viste.

Come precedentemente anticipato, ricorrere all'utilizzo di Framework come APEX può risultare utile nel momento in cui vogliamo creare delle semplici web app. Utilizzando HTML[3], infatti, eseguire particolari operazioni, quali ad esempio il collegamento tra due pagine distinte può portare ad un lungo lavoro. Dovremmo infatti definire una gerarchia a livelli, e utilizzare dei link alle pagine richieste ogni volta che necessitiamo di una relazione tra gli elementi delle relative pagine.
Utilizzando frameworks come Oracle APEX, tramite semplici query che effettuano operazioni di join tra tabelle, possiamo facilmente accedere a tutti i dati di cui necessitiamo.

[3] HyperText Markup Language. Linguaggio di markup, nato per la formattazione di documenti ipertestuali. Viene utilzzato soprattutto per la realizzazione di pagine web.

Trattandosi di uno strumento che permette anche di trattare la persistenza dei dati, è opportuno dividere nella trattazione quelle che sono le funzionalità relative alla gestione di un Database, da quelle relative alla creazione di un'applicazione.

2.1 Gestione dei dati e creazione degli oggetti

2.1.1 Creazione di oggetti

APEX permette, tramite una serie di passaggi, la creazione di qualsiasi entità necessaria alla realizzazione di un Database. In seguito alla creazione, verrà mostrato dal sistema anche il codice SQL[4] utilizzato, messo in appendice

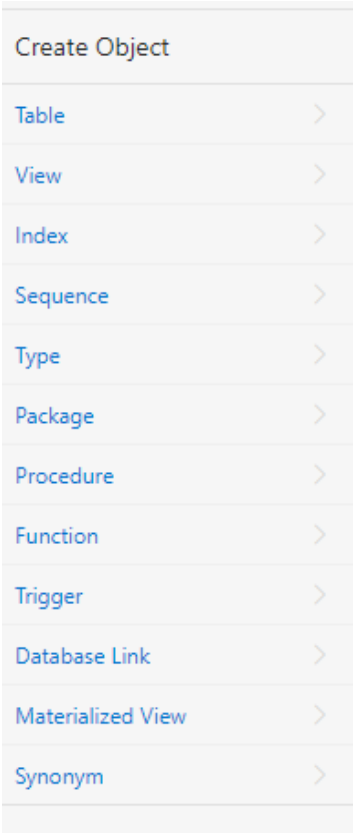


Figura 1 - Menu relativo agli oggetti creabili con APEX

[4] Structured Query Language. Si tratta di un linguaggio standardizzato per database basati sul modello relazionale (RDBMS)

2.1.2 Creazione di una tabella

L'elemento fondante di una base di dati è rappresentato da una tabella in cui vengono inserite le informazioni.

APEX permette, tramite una serie di passaggi, la creazione di queste tabelle.

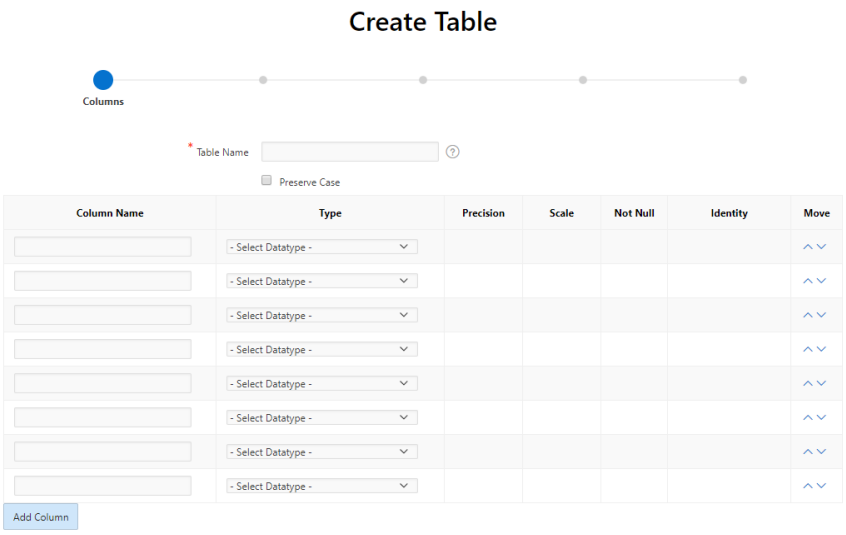


Figura 2 - Form per la creazione di una tabella

Come si può evincere dalla figura 2, l'interfaccia è molto semplice, e permette di inserire i vari campi, con il tipo, la scala e di spuntare una casella, per indicare se un elemento può essere, o meno, null.

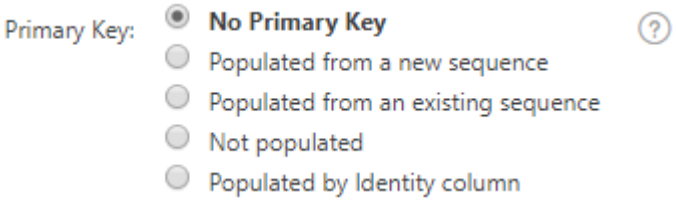


Figura 3 - Inserimento chiave primaria

Successivamente viene richiesta la scelta, ed eventualmente il campo corrispondente, di una chiave primaria, che come da definizione è rappresentato da un insieme univoco di *tuple*. Generalmente si riferisce ad un campo identificativo in particolare della tabella stessa.

Possiamo inoltre aggiungere dei *constraint*, cioè nient'altro che vincoli ai campi della nostra tabella, specificando quelle che sono eventuali chiavi esterne(campi della tabella sono referenziati con altri di tabelle diverse), o semplici requisiti organizzativi, quali ad esempio la lunghezza minima e/o massima di un eventuale stringa password.

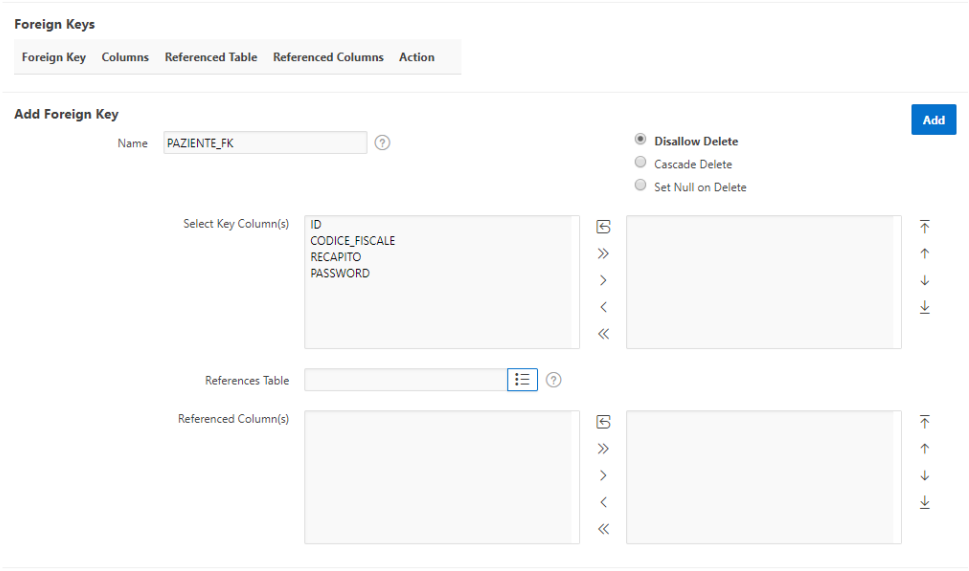


Figura 4 - Inserimento eventuali chiavi esterne

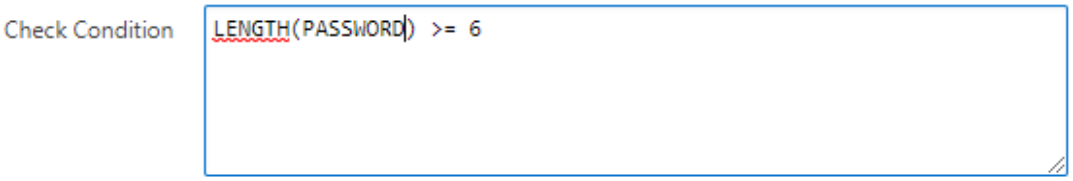


Figura 5 - Condizione sul campo password

2.1.3 Creazione di un Trigger

Nelle basi di dati, il trigger è una procedura eseguita automaticamente dal sistema, ogni qualvolta si verifica un determinato evento(delete, insert o update di una linea all'interno di una tabella). I trigger ci permettono quindi di specificare e mantenere complessi vicoli d'integrità.

Possiamo fare una differenza tra i trigger a livello riga, cioè quelli che agiscono " *for each row*", e quelli che agiscono a livello dell'istruzione completa.

La differenza sostanziale tra i due tipi di trigger è rappresentata dal fatto che mentre un trigger a livello di riga viene eseguito una volta per ogni riga, in base al compiersi dell'evento che l'ha scatenato, un trigger a livello di istruzione viene eseguito una e una sola volta, indipendentemente dal numero di righe alle quali ci stiamo riferendo.

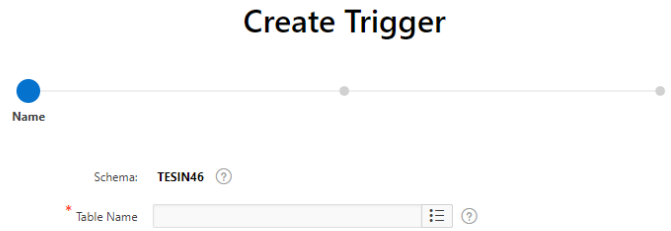


Figura 6 - Creazione trigger

La figura 6 mostra una semplice interfaccia tramite la quale possiamo inserire la tabella alla quale il trigger farà riferimento.

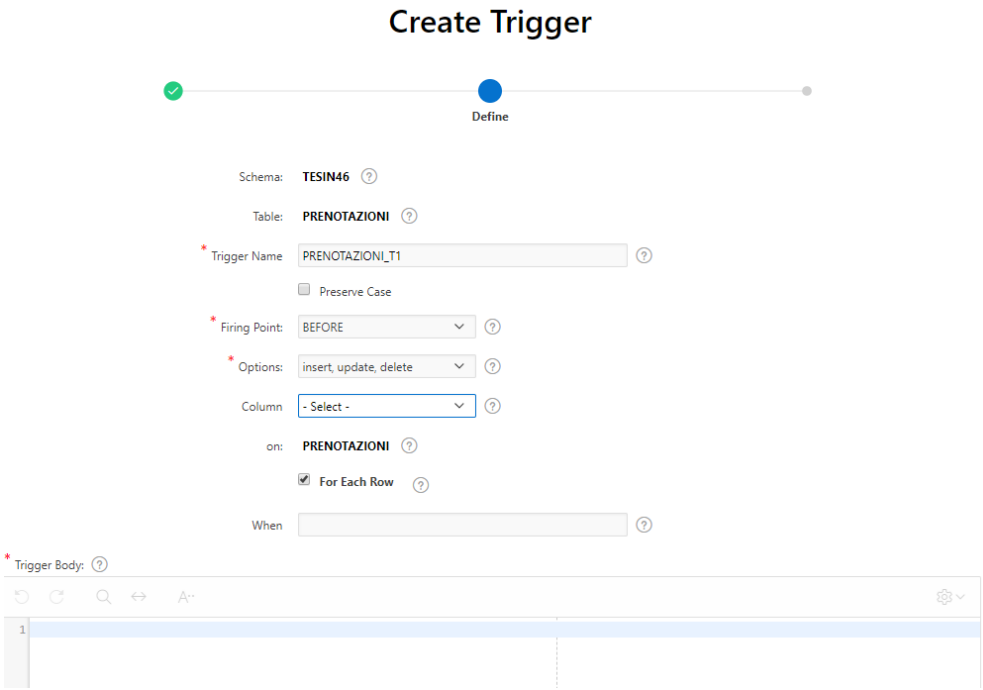


Figura 7 - Inserimento campi per il corpo del trigger

Dalla figura 7 si evince ancora una volta la semplicità con la quale APEX permette di eseguire operazioni. Possiamo infatti banalmente riempire i campi richiesti, e, usando un linguaggio PL/SQL[5], definire il corpo del trigger, che verrà in seguito creato.

[5] Procedural Language/Structured Query Language. Linguaggio di programmazione proprietario per database di Oracle Corporation, procedurale. È un estensione dell'SQL.

2.1.4 Creazione di una vista materializzata

Nelle basi di dati, utilizziamo le viste per semplificare l'utilizzo delle query[6]. Per evitare infatti un eccessivo uso di operazioni di Join tra tabelle, si preferisce utilizzare questa tecnica, che realizza una sintesi basata sull'unione dei dati di diverse tabelle.

Le viste materializzate, differiscono da quelle standard, in quanto vengono scritte direttamente sul disco, al fine di ottenerne un accesso più rapido. Il loro contenuto viene inoltre periodicamente aggiornato tramite il DBMS.

"Tutte le viste che sono aggiornabili in teoria, dovrebbero esserlo anche in pratica".[1]

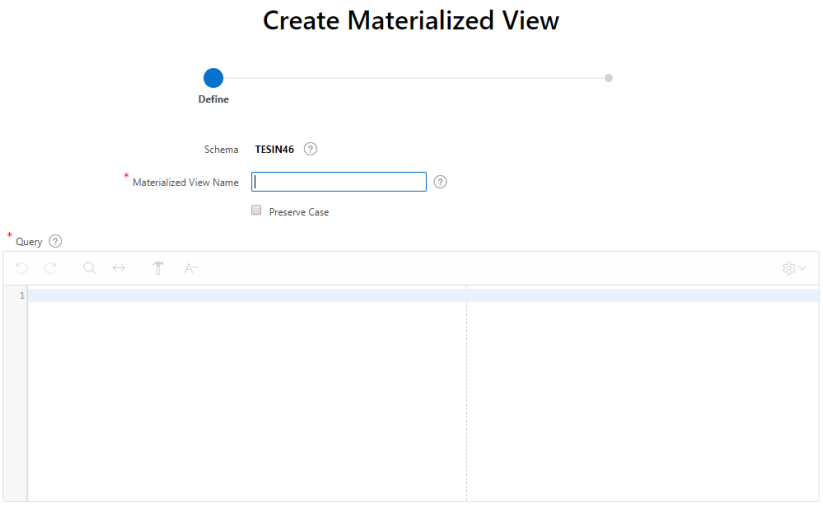


Figura 8 - Creazione di una vista materializzata

Il form di creazione della vista materializzata(Figura 8) permette tramite l'inserimento del codice relativo alla query da implementare, la realizzazione della suddetta vista. APEX mette inoltre a disposizione il tool "*Query builder*", tramite il quale viene messa a disposizione la scelta di una o più tabelle, sulle quali generare lo script della query desiderata.

[6] Interrogazione di un Database

2.2 Creazione dell’applicazione

2.2.1 Inizializzazione dell’app

La prima fase relativa alla realizzazione di un’applicazione web, mediante l’uso di APEX, prevede la creazione di quella che è la struttura generica dell’app stessa.

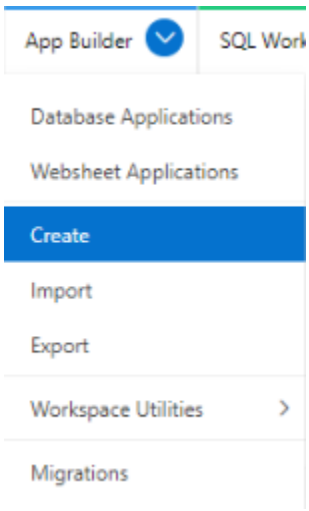


Figura 9 - Creazione App

La successiva schermata(Figura 10) ci offre numerose funzionalità, tra le quali, oltre la semplice scelta del nome dell’applicazione, anche quella di aggiunta di pagine. Possiamo inoltre selezionare alcune features ed opzioni per il nostro progetto.

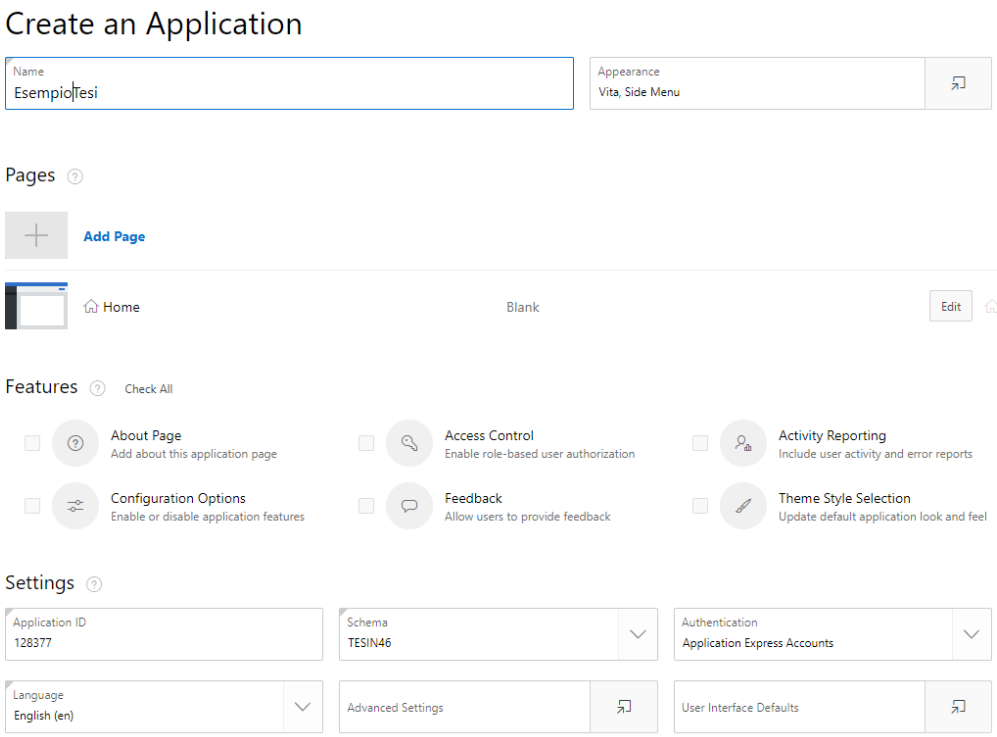


Figura 10 - Form di creazione di un applicazione

Features offerte:

- About Page: Inserisce una descrizione dell'app, inlcudendone la versione e il numero di pagine presenti.
- **Access Control**: Definisce dei ruoli all'interno dell'applicazione. Gli utenti possono ricevere permessi di amministrazione, moderazione o semplice lettura.
- Activity reporting: Vengono inclusi dei report riguardanti l'attività dell'app, e dei relativi utenti. In questi report troviamo anche il numero degli errori commessi dai vari utenti, che saranno utili a migliorare l'apprendibilità del sistema.
- Configuration Options: Tramite questa feature, gli admin possono abilitare o disabilitare specifiche funzionalità dell'applicazione.

- Feedback: Inserisce un meccanismo di feedback, tramite il quale gli utenti possono postare commenti relativi all'applicazione. Questi possono includere anche informazioni sulla sessione relative all'utente segnalante.
- Theme style selection: Permette all'utente di selezionare temi predefiniti per l'applicazione.

2.2.2 Aggiunta di pagine

La funzionalità fondamentale di questo tool è sicuramente l'aggiunta di nuove pagine per l'applicazione.

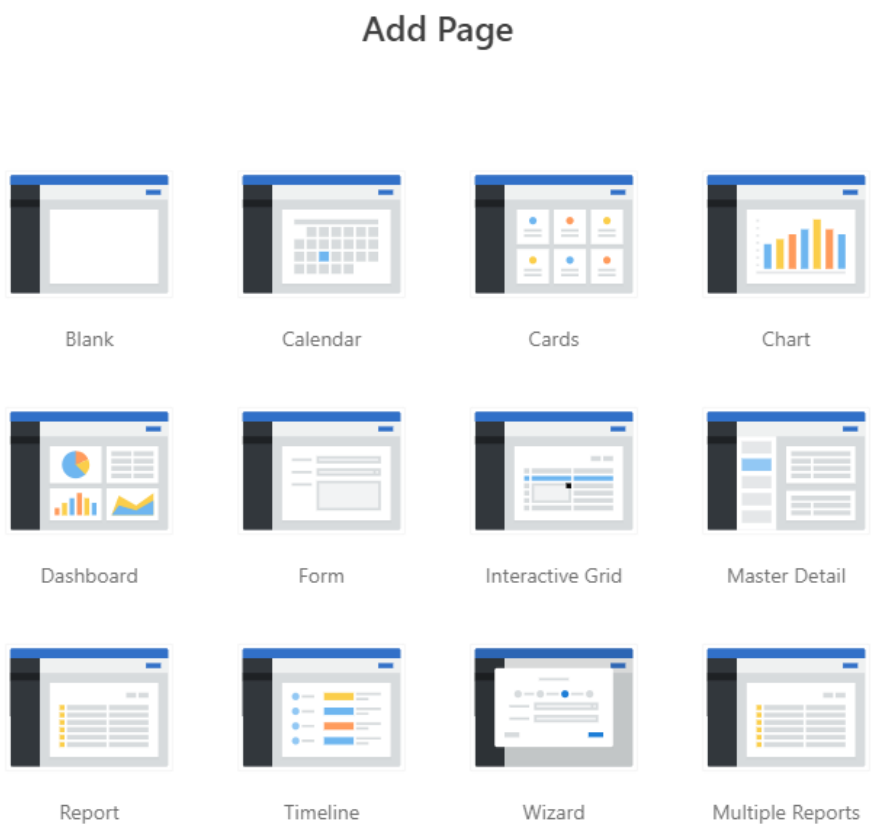


Figura 11 - Aggiunta di pagine

Possiamo, come si evince dalla figura 11, selezionare vari tipi di pagine, che spaziano dai grafici ai calendari.

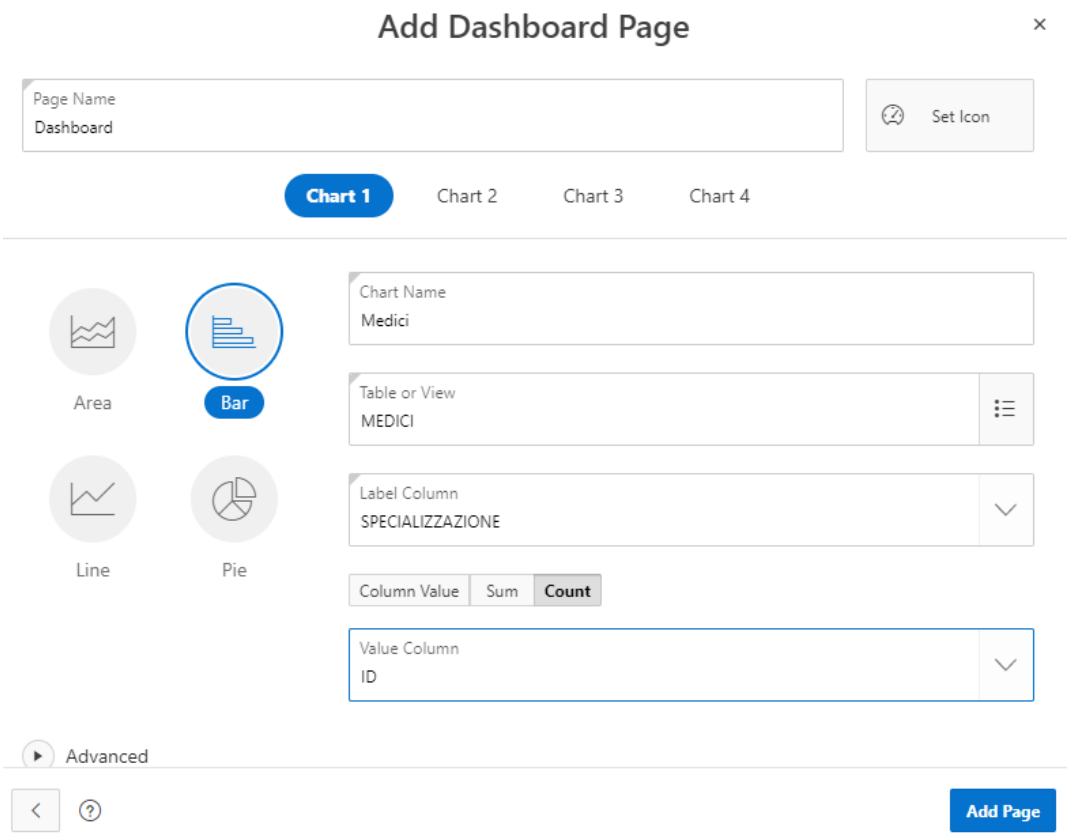


Figura 12 - Creazione di una dashboard

La dashboard prevede l'inserimento di dei grafici(da 1 a 4). Come si nota in Figura 12, APEX offre 4 tipi di grafici(ad area, a barre, a torta o lineare), che proietteranno i valori, le somme di questi, o ancora conteranno le occorrenze, di una tabella che possiamo selezionare dall'apposito form.

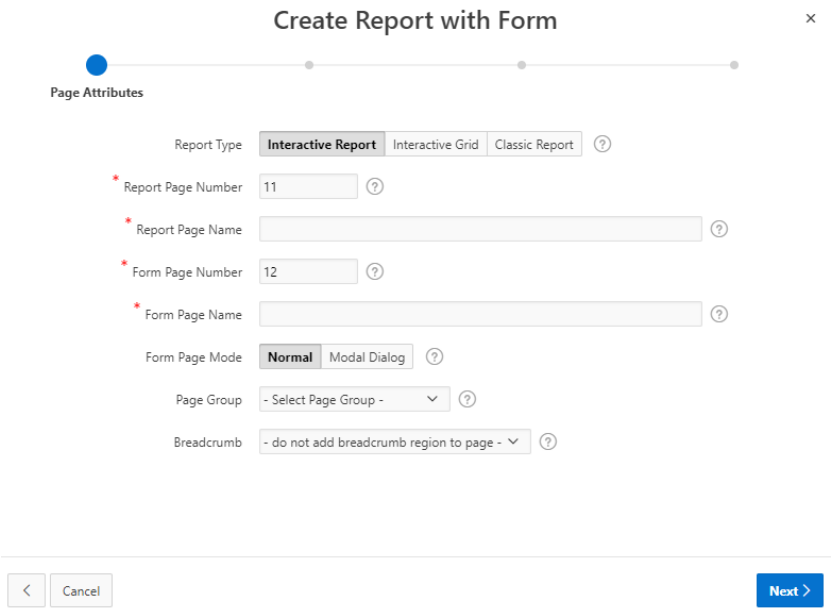


Figura 13 - Creazione di un report con form

Possiamo inoltre creare un report con un relativo form di inserimento. Nel report verrà mostrata una lista di occorrenze, relative ai campi della tabella che sceglieremo in fase di creazione.

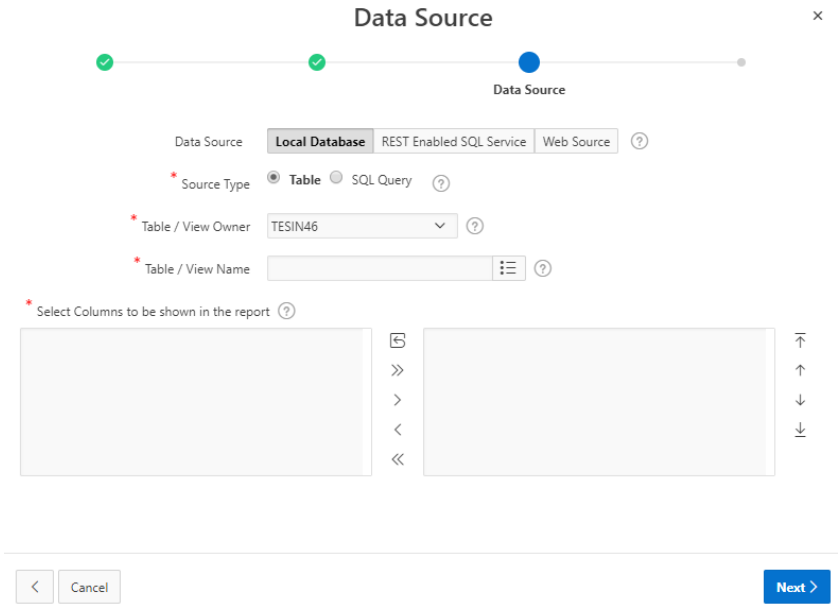


Figura 14 - Scelta delle tabelle per il report

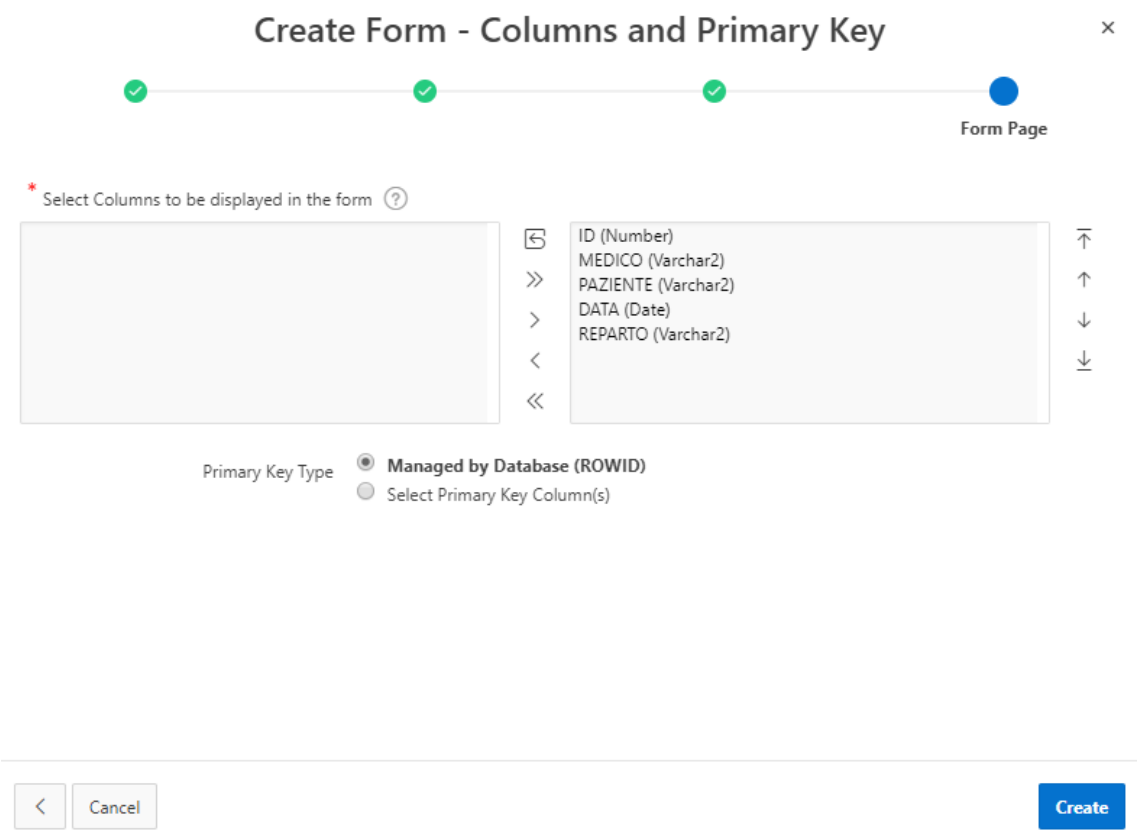


Figura 15 - Creazione form relativo al report

Per quanto riguarda, invece, il form, troviamo all'interno della procedura di creazione del report, un'ultima sezione nella quale possiamo scegliere i campi della tabella che comporranno il modulo di inserimento.

La pagina appena creata, può essere inoltre modificata direttamente. Tramite l'apposito editor, possiamo infatti gestire i singoli componenti, e le loro funzionalità. Si può ad esempio decidere di popolare un menu a tendina con tutti i dati presenti in un campo della tabella selezionata in precedenza.

Un ultimo tipo di pagina che andremo ad analizzare è quella del calendario. Verrà creato un vero e proprio calendario, nel quale saranno segnate giorno per giorno(è richiesto un attributo di tipo *Date* nella tabella selezionata) le relative occorrenze.

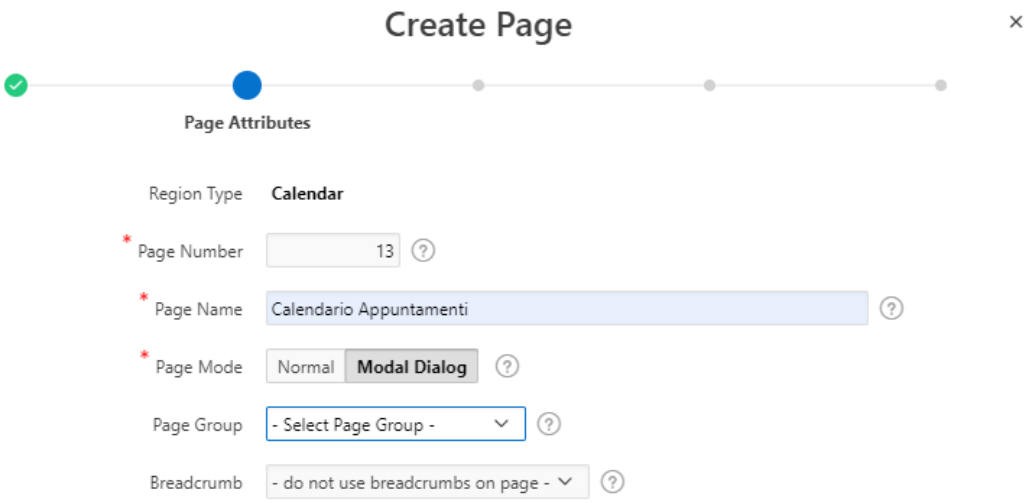


Figura 16 - Creazione del calendario

Oltre alla solita scelta della tabella e dei campi relativi, possiamo inserire la colonna da mostrare sul riquadro del giorno

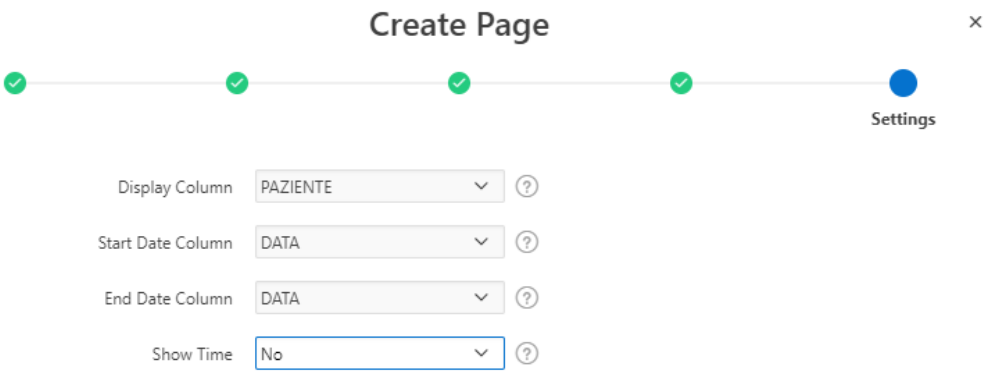


Figura 17 - Scelta dei campi da mostrare nel calendario

Come anticipato, la funzionalità *edit page* prevede, tra le altre cose, l'inserimento di elementi ulteriori, quali aree di testo compilabili o menu a tendina.(Figura 18)

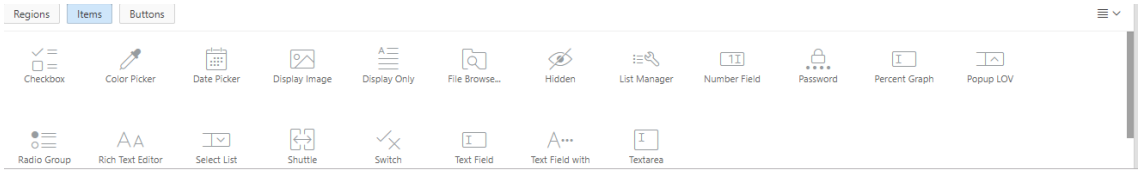


Figura 18 - Elementi aggiuntivi disponibili in APEX

Come esempio(in Figg. 19 e 20) viene considerato l'aggiunta di un'area di testo modificabile, che in base al contenuto e alla query definita, va a modificare l'aspetto del calendario.

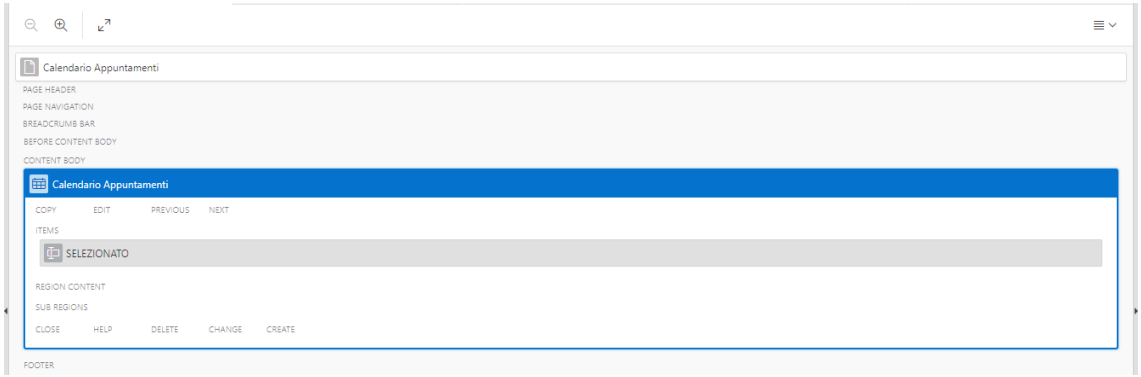


Figura 19 - Aggiunta di un elemento

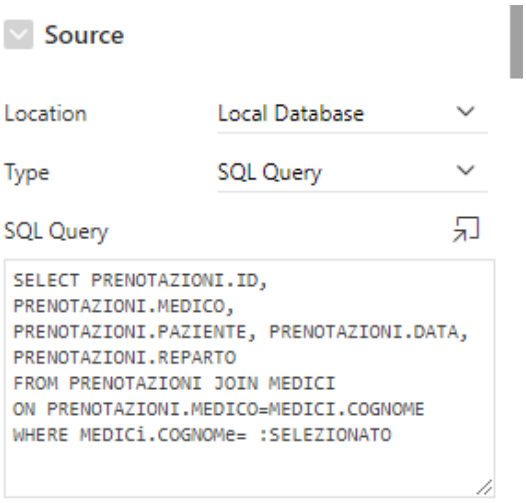


Figura 20 – Spazio designato per la query relativa all'elemento aggiunto

Capitolo 3 : Esempio di applicazione

In questo capitolo verrà illustrato, tramite un esempio, il funzionamento effettivo dello strumento in analisi: Oracle APEX. Al fine della trattazione, utilizzando le funzionalità già mostrate in precedenza, verrà utilizzato un database popolato con dei dati di prova.

3.1 Scelte progettuali

Verrà preso in esempio, al fine di ottenere una trattazione semplice e sicuramente non performante, un database elementare, che simulerà la persistenza e la gestione di dati relativi a pazienti, medici e relative prenotazioni di visite mediche in un ospedale da parte di un operatore esterno, che è interessato a visualizzare, modificare e inserire nuove prenotazioni in memoria.

Il modello ER[7] che descrive la base di dati presa in esame è il seguente

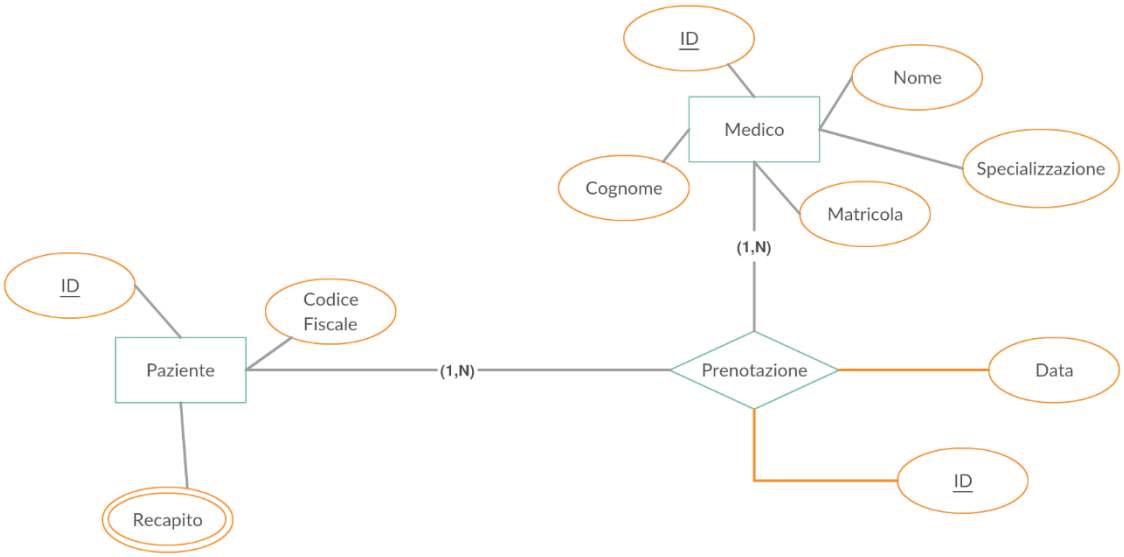


Figura 21 - Modello ER

3.2 Avvio dell'applicazione

Al momento dell'avvio dell'applicazione, ci verrà chiesto di effettuare l'accesso utilizzando i dati, con i quali si è preventivamente compiuta la procedura di registrazione alla piattaforma.

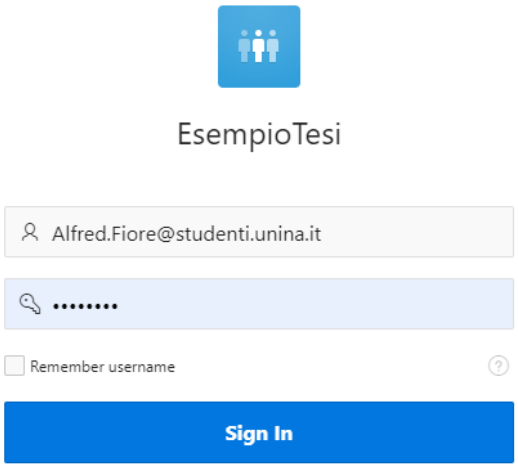


Figura 22 - Form di login

[7]Modello Entità-Relazione. Si tratta di un modello teorico, utilizzato per rappresentare concettualmente e graficamente dei dati, mantenendo un alto livello di astrazione.

A questo punto, ci troveremo davanti alla schermata iniziale dell'applicazione, che come da decisione presa al momento della creazione, mostrerà dei riferimenti alle varie pagine.

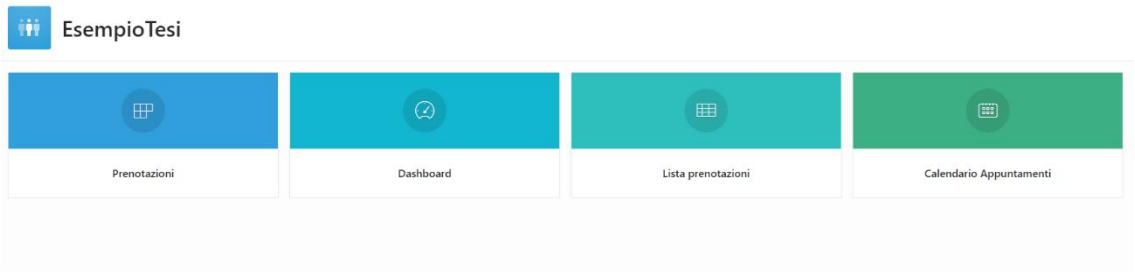


Figura 23 - Home page dell'applicazione

3.3 Funzionalità implementate

Come anticipato nel Capitolo 2, APEX permette la creazione di svariati tipi di pagine. Per semplificare la trattazione e mostrare solo alcune di quelle che sono le enormi potenzialità che questo strumento offre, si è deciso di implementare solamente le seguenti funzionalità:

3.3.1 Grafici relativi ai medici e alle prenotazioni

In questa prima sottosezione, andiamo ad analizzare quella che è la *Dashboard* dell'applicazione. In cui vengono presentati due grafici. Il primo, a destra in figura 24, mostra un grafico a barre rappresentante il numero di medici per relativa specializzazione, afferenti all'ospedale.

Il secondo, a sua volta ci da un'informazione di quante prenotazioni sono state effettuate per i relativi reparti ospedalieri.

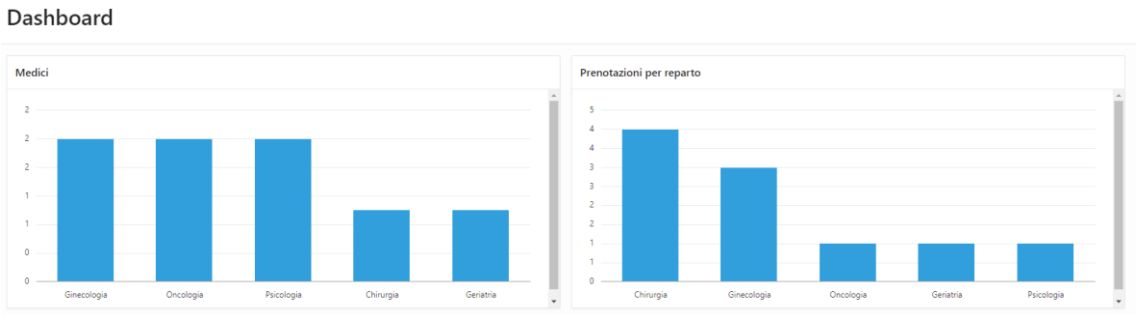
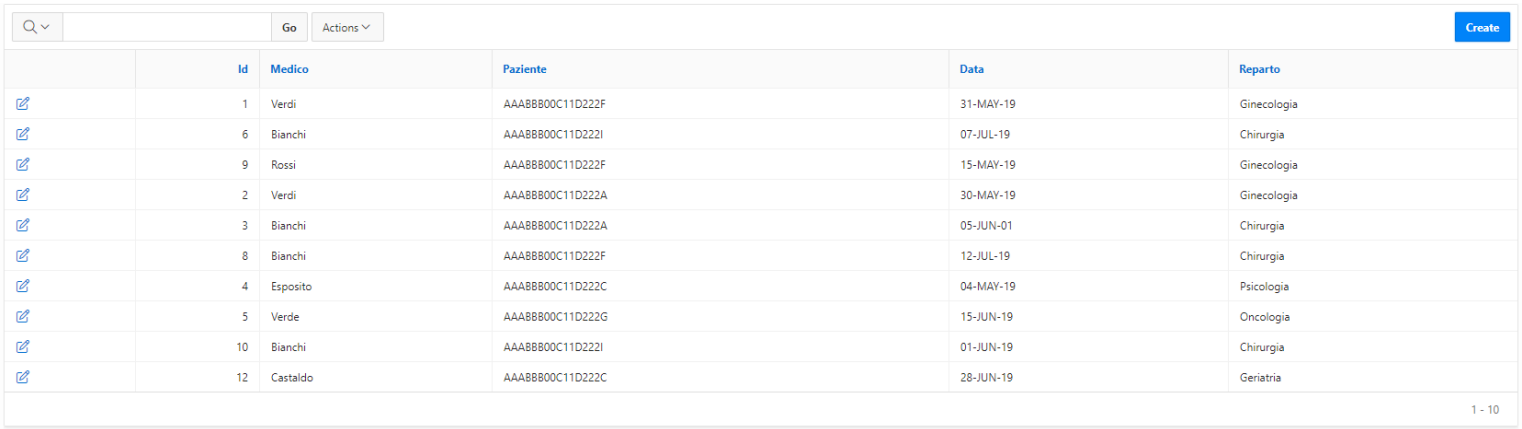


Figura 24 - Dashboard

3.3.2 Lista Prenotazioni e relativo form di inserimento

In questa pagina, troviamo un report con il proprio form di creazione.

In particolare, come si nota dalla figura 25, è presente una lista di tutte le prenotazioni effettuate da un qualsiasi paziente, con un relativo medico. Viene inoltre specificata la data e ovviamente il reparto al quale la prenotazione fa riferimento.



	Id	Medico	Paziente	Data	Reparto
	1	Verdi	AAABBB00C11D222F	31-MAY-19	Ginecologia
	6	Bianchi	AAABBB00C11D222I	07-JUL-19	Chirurgia
	9	Rossi	AAABBB00C11D222F	15-MAY-19	Ginecologia
	2	Verdi	AAABBB00C11D222A	30-MAY-19	Ginecologia
	3	Bianchi	AAABBB00C11D222A	05-JUN-01	Chirurgia
	8	Bianchi	AAABBB00C11D222F	12-JUL-19	Chirurgia
	4	Esposito	AAABBB00C11D222C	04-MAY-19	Psicologia
	5	Verde	AAABBB00C11D222G	15-JUN-19	Oncologia
	10	Bianchi	AAABBB00C11D222I	01-JUN-19	Chirurgia
	12	Castaldo	AAABBB00C11D222C	28-JUN-19	Geriatria

1 - 10

Figura 25 - Lista prenotazioni presenti in memoria

Per quanto concerne, invece, il form di creazione(attivabile cliccando il tasto create in alto), troviamo un'interfaccia molto semplice, nella quale oltre ad inserire un id numerico, possiamo selezionare tramite menu a tendina sia il codice fiscale dei pazienti che sono registrati all'ospedale(ai fini della trattazione si presuppone che questi abbiano compilato un modulo di accettazione all'ospedale), e del relativo medico con il quale vogliamo effettuare la visita. Per quanto riguarda invece la data della suddetta prenotazione, possiamo sia inserirla manualmente, che utilizzando un calendario interattivo messo a disposizione da APEX.

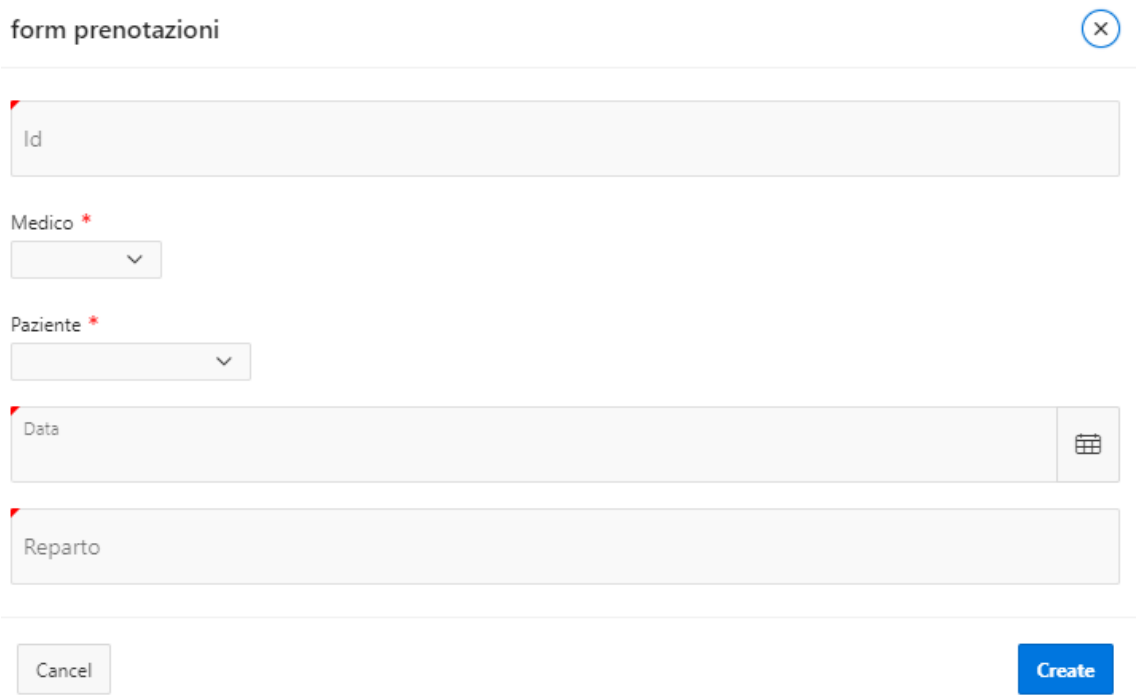


Figura 26 - Form Prenotazioni

Altre funzionalità disponibili in questa pagina, sono per esempio, la ricerca delle occorrenze relative ad un certo medico, o paziente, effettuabile semplicemente inserendo una stringa di testo, che il sistema, tramite apposite query provvederà a *matchare* con i dati presenti all'interno del database.

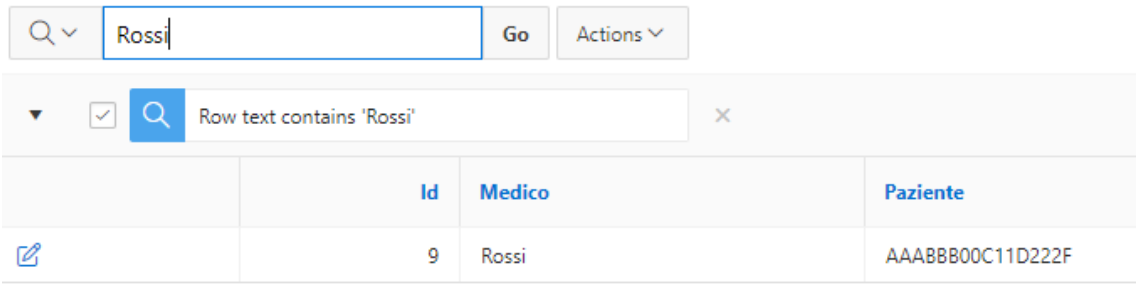


Figura 26 - Ricerca nella lista

3.3.3 Descrizione Generale delle prenotazioni

Viene implementata in questa sezione la pagina di tipo *Cards*, che fornisce, in una veste grafica alquanto gradevole, un *outline* delle prenotazioni presenti in memoria. Vengono infatti mostrate tutte le info relative ad esse. Si è scelto di visualizzare come campo principale la data.

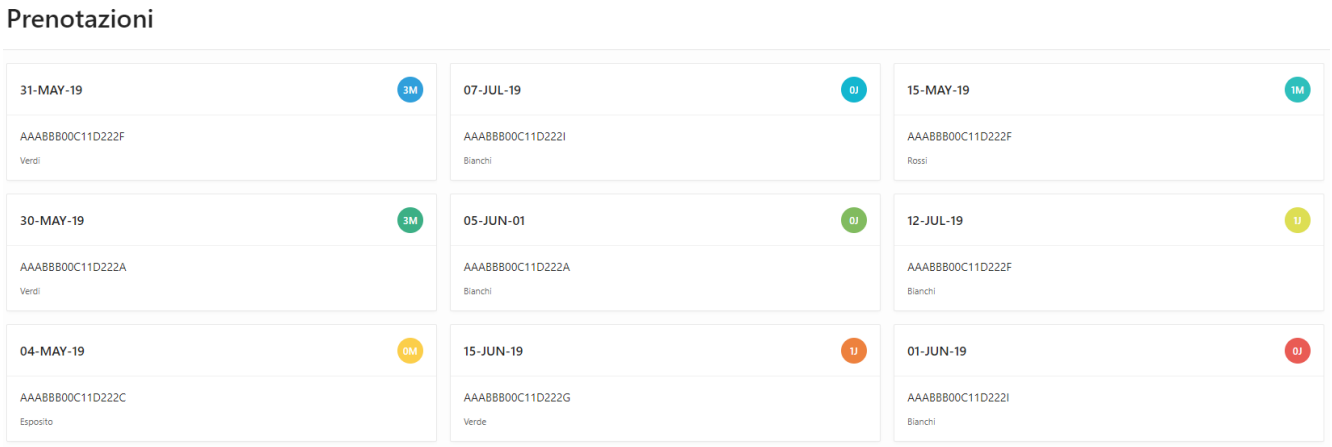


Figura 27 – Cards

Come ovvio che sia, un qualsiasi nuovo inserimento di una prenotazione all'interno del sistema, porterà ad un aggiornamento automatico della pagina.

3.2.4 Calendario Appuntamenti

L'ultima funzionalità implementata che si è voluta prendere in esame, è quella relativa alla realizzazione di un calendario, riportante, per ogni giorno in cui è presente una prenotazione, il codice fiscale del paziente prenotato.

Viene inoltre inserito all'interno della pagina, un campo di testo vuoto, che se riempito con il nome di un medico, dopo la pressione del tasto *Invio*, va a modificare il calendario mostrando solo gli appuntamenti del medico selzionato. Si è scelto, per snellire la trattazione, pur restando in un caso più che verosimile, di consentire l'inserimento di più prenotazioni per ogni giorno.

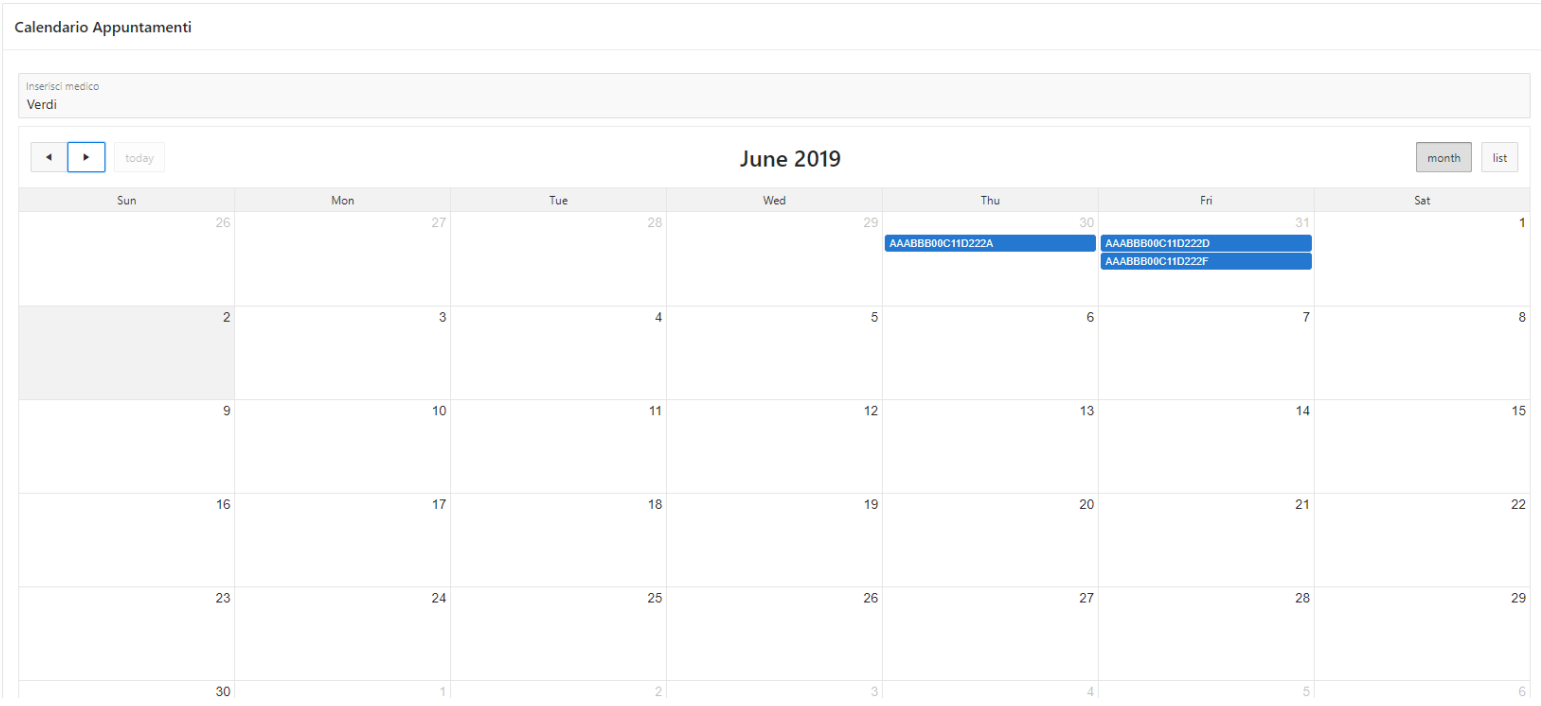


Figura 27 - Calendario appuntamenti del medico Verdi

Conclusioni

Come già precedentemente sottolineato, l'applicazione rappresenta solo una minima parte di quelle che sono le tantissime funzionalità e le enormi potenzalità del tool Oracle APEX.

Si è voluto comunque, in questo elaborato, tramite esempi di quelle che sono gli strumenti essenziali messi a disposizione dall'applicazione, far notare la semplicità e l'intuitività del tool. Scrivendo pochissime linee di codice, e realizzando solo un modello ER, si è riusciti infatti a realizzare una web app, che rispondesse ad input dell'utente, andando a gestire inserimenti, cancellazioni, e modifiche di dati salvati all'interno di un database realizzato mediante APEX stesso.

Sicuramente, utilizzare un software del genere per creare delle applicazioni web risulta molto più semplice rispetto alla realizzazione di queste ultime in linguaggi come JAVA o HTML.

Sviluppi possibili dell'applicazione realizzata, possono sicuramente essere rappresentati da una gestione diversa delle prenotazioni, la quale tabella tramite operazioni relative al DBMS potrebbe periodicamente aggiornarsi.

Appendice

A1 Creazione delle tabelle e definizione dei vincoli di integrità referenziale

```
CREATE TABLE  "PAZIENTE"(
        “ID" NUMBER(2,0) NOT NULL ENABLE,
        "CODICE_FISCALE" VARCHAR2(20) NOT NULL ENABLE,
        "RECAPITO" VARCHAR2(10) NOT NULL ENABLE,
         CONSTRAINT "PAZIENTE_PK" PRIMARY KEY ("ID")
        )
CREATE table "MEDICI" (
  "ID" NUMBER(2) NOT NULL,
   "NOME" VARCHAR2(20) NOT NULL,
   "COGNOME" VARCHAR2(20) NOT NULL,
   "MATRICOLA" VARCHAR2(20) NOT NULL,
   "SPECIALIZZAZIONE" VARCHAR2(20) NOT NULL,
   constraint  "MEDICI_PK" primary key ("ID")
)
CREATE table "PRENOTAZIONI" (
   "ID" NUMBER(2) NOT NULL,
   "DATA" DATE NOT NULL,
   "PAZIENTE" VARCHAR2(20) NOT NULL,
   "MEDICO" VARCHAR2(20) NOT NULL,
   constraint  "PRENOTAZIONI_PK" primary key ("ID")
)
/
```

```
ALTER TABLE "PRENOTAZIONI" ADD CONSTRAINT "PRENOTAZIONI_FK"
FOREIGN KEY ("PAZIENTE")
REFERENCES "PAZIENTI" ("CODICE_FISCALE")
ON DELETE SET NULL;
ALTER TABLE "PRENOTAZIONI" ADD CONSTRAINT "PRENOTAZIONI_FK2"
FOREIGN KEY ("MEDICO")
REFERENCES "MEDICI" ("COGNOME")
ON DELETE SET NULL;
```

A2 Popolamento delle tabelle

A2.1 Tabella Pazienti

```
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222A’, ‘3330000001’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222B’,’3330000002’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222C’,’3330000003’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222D’,’3331110002’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222E’,’3339270173’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222F’,’3459163957’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222G’,’3829573492’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222H’,’3459163967’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222I’,’3459443957’);
INSERT INTO PAZIENTI VALUES (‘1’, ‘AAABBB00C11D222L’,’3459162457’);
```

A2.2 Tabella Medici

INSERT INTO MEDICI VALUES (‘1’, ‘Mario, ‘Rossi’,Mat001,’Ginecologia’);

INSERT INTO MEDICI VALUES (‘2’, ‘Mario’, ‘Verdi’,Mat002,’Ginecologia’);

INSERT INTO MEDICI VALUES (‘3’,‘Alessandro’,‘Neri’,’Mat003’,’Oncologia’);

INSERT INTO MEDICI VALUES (‘4’, ‘Antonio’, ‘Castado’,Mat004,’Geriatria’);

INSERT INTO MEDICI VALUES (‘5’, ‘Giovanni’, ‘Verde’,Mat005,’Chirurgia’);

INSERT INTO MEDICI VALUES (‘6’, ‘Alfredo’, ‘Bianchi’,Mat006,’Chirurgia’);

INSERT INTO MEDICI VALUES (‘7’, ‘Carla’, ‘Esposito’,Mat007,’Psicologia’);

INSERT INTO MEDICI VALUES (‘8’, ‘Carla’, ‘De Rossi’,Mat008,’Psicologia’);

A2.3 Tabella Prenotazioni

INSERT INTO PRENOTAZIONI VALUES (‘1, ‘05/31/2019’, ‘AAABBB00C11D222F’,’Verdi’,’Ginecologia’);

INSERT INTO PRENOTAZIONI VALUES (‘2, ‘05/30/2019’, ‘AAABBB00C11D222A’,’Verdi’,’Ginecologia’);

INSERT INTO PRENOTAZIONI VALUES (‘3, ‘07/31/2019’, ‘AAABBB00C11D222A’,’Bianchi’,’Chirurgia’);

INSERT INTO PRENOTAZIONI VALUES (‘4, ‘04/05/2019’, ‘AAABBB00C11D222C’,’Esposito’,’Psicologia’);

INSERT INTO PRENOTAZIONI VALUES (‘5, ‘06/15/2019’, ‘AAABBB00C11D222G’,’Verde’,’Oncologia’);

A3 Queries omesse nella trattazione

Per la realizzazione della funzionalità mostrata in Figura 27, viene utlizzata una query che, effettuando un operazione di join tra tabella “Prenotazioni” e la tabella “Medici”, restituisce, e quindi mostra sul calendario, solamente le prenotazioni effettuate con il medico scelto. La query utilizzata è la seguente:

```
SELECT PRENOTAZIONI.ID,PRENOTAZIONI.MEDICO, PRENOTAZIONI.PAZIENTE,
PRENOTAZIONI.DATA, PRENOTAZIONI.REPARTO
FROM PRENOTAZIONI JOIN MEDICI
ON PRENOTAZIONI.MEDICO=MEDICI.COGNOME
WHERE MEDICi.COGNOME= :SELEZIONATO[8]
```

Per il popolamento dei menu a tendina, invece le query utilizzate sono banalmente le seguenti:

```
SELECT COGNOME
FROM MEDICI
```

```
SELECT CODICEFISCALE
FROM PAZIENTI
```

[8]L’attributo “Selezionato” non è altro che il nome del componente aggiuntivo Text Field, nel quale verrà scritto il nome del medico

Bibliografia

[1] sesta legge formale : https://it.wikipedia.org/wiki/Vista_(basi_di_dati)