

Centr Vista

- CentroVista S.r.l. è un'azienda che fornisce servizi in ambito sanitario e nello specifico opera nel campo dell'oftalmologia.
- Con una presenza sul territorio molto importante (cinque sedi attualmente) e un'ampia schiera di specialisti è ormai una delle realtà più importanti e affermate nel territorio umbro.

Documentazione dei requisiti

- Un sistema che consideri l'esistenza di due tipologie di utenti che possono utilizzare il servizio : Dottori (interni all'azienda) e i clienti.
- Il sistema da implementare avrà come scopo la possibilità di far prenotare ai clienti una visita in uno dei centri preposti, dando loro la possibilità di scegliere il medico da cui farsi visitare e in quale fascia oraria.
- I dottori invece potranno gestire i propri orari e consultare gli appuntamenti a loro rivolti.

Specifiche di implementazione

- Utilizzo di un database per contenere i dati relativi al sistema.
- Utilizzo di phpMyAdmin per una comoda gestione della base di dati.



- Creazione di una API REST per consentire l'interazione con la base di dati da parte di un qualsiasi sistema software che voglia in futuro interfacciarsi con i dati dell'azienda e inglobare i servizi di prenotazione messi a disposizione nel loro programma. (ipotesi di creazione di una web app). Il linguaggio scelto è PHP.
- Implementazione di una mobile application creata utilizzando l'IDE Android Studio e il linguaggio Kotlin




























Database

Per la rappresentazione delle risorse è stato scelto di creare cinque tabelle in MySql raffiguranti le entità cruciali.

- 1) user <--- clienti con tutti i loro dati
- 2) doctor <--- dottori con tutti i loro dati
- 3) branches <--- sedi con tutti i loro dati
- 4) timeslot <--- fasce orarie di ogni dottore
- 5) appointment <--- appuntamenti registrati

1. user

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
<input type="checkbox"/> 1	cdf 	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 2	name	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 3	surname	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 4	gender	varchar(2)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 5	dateOfBirth	date			No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 6	email	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 7	phoneNumber	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 8	passwd	varchar(100)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più










- Il codice fiscale viene usato come chiave primaria in quanto è un identificativo per natura univoco.

2. doctor

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
<input type="checkbox"/> 1	doctorId 	int			No	Nessuno		AUTO_INCREMENT	 Modifica  Elimina  Più
<input type="checkbox"/> 2	name	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 3	surname	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 4	gender	varchar(2)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 5	dateOfBirth	date			No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 6	email	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 7	phoneNumber	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 8	headOffice	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 9	passwd	varchar(100)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più


- Il doctorId è un identificatore rilasciato all'atto dell'assunzione di un nuovo dottore e dunque scelto volutamente univoco.

3. branches

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
<input type="checkbox"/> 1	idBranch 	int			No	Nessuno		AUTO_INCREMENT	 Modifica  Elimina ▼ Più
<input type="checkbox"/> 2	address	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina ▼ Più
<input type="checkbox"/> 3	city	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina ▼ Più
<input type="checkbox"/> 4	switchboardNumber	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina ▼ Più























- In aggiunta ai dati fondamentali (indirizzo e città) vi è il numero di centralino per contattare la segreteria della struttura.

4. timeslot

	#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
<input type="checkbox"/>	1	doctor_id 	int			No	Nessuno
<input type="checkbox"/>	2	slot1	time			Si	NULL
<input type="checkbox"/>	3	slot2	time			Si	NULL
<input type="checkbox"/>	4	slot3	time			Si	NULL
<input type="checkbox"/>	5	slot4	time			Si	NULL
<input type="checkbox"/>	6	slot5	time			Si	NULL
<input type="checkbox"/>	7	slot6	time			Si	NULL
<input type="checkbox"/>	8	slot7	time			Si	NULL
<input type="checkbox"/>	9	slot8	time			Si	NULL
<input type="checkbox"/>	10	slot9	time			Si	NULL
<input type="checkbox"/>	11	slot10	time			Si	NULL
<input type="checkbox"/>	12	slot11	time			Si	NULL
<input type="checkbox"/>	13	slot12	time			Si	NULL
<input type="checkbox"/>	14	slot13	time			Si	NULL
<input type="checkbox"/>	15	slot14	time			Si	NULL

- Per scelte aziendali tutte le strutture fanno orario continuato e ai dottori che vi lavorano viene data la possibilità di definire 14 orari di visita durante la giornata nei quali è possibile ricevere i clienti con una distribuzione delle visite flessibile. Infatti la scelta degli orari è libera e a discrezione dei dottori.

5. appointment

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito	Commenti	Extra	Azione
<input type="checkbox"/> 1	appointmentId 	varchar(10)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 2	cdf 	varchar(255)	utf8mb4_0900_ai_ci		No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 3	doctorId 	int			No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 4	branchId 	int			No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 5	day	date			No	Nessuno			 Modifica  Elimina  Più
<input type="checkbox"/> 6	timeSlot	time			No	Nessuno			 Modifica  Elimina  Più

Ogni appuntamento è caratterizzato da:

- Un appointmentId univoco casuale creato al momento di una prenotazione avvenuta con successo (servirà al cliente poi per poter modificare lo stesso) .
- Codice fiscale cliente.
- Identificatore del dottore.
- Sede.
- Giorno.
- Ora.

API REST

- È il cuore di questo progetto. Grazie alla sua implementazione Centro Vista sarà in grado di rilasciare i suoi servizi a chiunque li richieda in un qualsiasi ambiente software (sito web, applicazione mobile, programma desktop).
- REST si basa sulla definizione di endpoint URI i quali verranno opportunamente chiamati tramite il protocollo http nei codici sorgente dei programmi che ne vorranno fare uso. Gli endpoint come risposta rilasceranno delle risorse da loro gestite, tipicamente in formato JSON, dando così la possibilità al client di elaborarle come meglio crede.

API REST (esempio)

```
// READ doctor
function read()
{
    // select all
    $query = "SELECT
                *
            FROM
                " . $this->table_name . ",branches where headOffice=idBranch";
    $stmt = $this->conn->prepare($query);
    // execute query
    $stmt->execute();
    return $stmt;
}
```

- Questa funzione viene eseguita quando si effettua una richiesta al seguente uri:
http://centrovista.it/centro_oculisticoREST/doctor/read.php

- Semplice query all'interno della funzione read() nella classe Dottori. Ritorna un oggetto contenente il risultato dell'interrogazione.

API REST (esempio cont.)

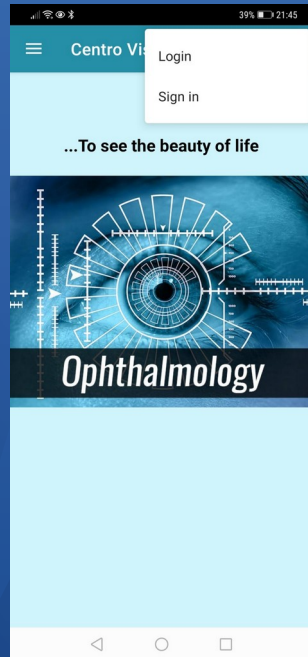
```
header( header: "Access-Control-Allow-Origin: *");
header( header: "Content-Type: application/json; charset=UTF-8");
// includiamo database.php e branch.php per poterli usare
include_once '../config/database.php';
include_once '../models/doctor.php';
// creiamo un nuovo oggetto Database e ci colleghiamo al nostro database
$database = new Database();
$db = $database->getConnection();
// Creiamo un nuovo oggetto doctor
$doctor = new Doctor($db);
// query products
$stmt = $doctor->read();
$num = $stmt->rowCount();
// se vengono trovati dottori nel database
if($num>0){
    // array di dottori
    $doctor_arr = array();
    $doctor_arr = array();
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
        extract( &array: $row);
        $doctor_item = array(
            "doctorId" => $doctorId,
            "name" => $name,
            "surname" => $surname,
            "dateOfBirth" => $dateOfBirth,
            "gender" => $gender,
            "email" => $email,
            "phoneNumber" => $phoneNumber,
            "headOffice" => $address.", ".$city,
            "passwd" => $passwd
        );
        array_push( &array: $doctor_arr, $doctor_item);
    }
    http_response_code( response_code: 200);
    echo json_encode($doctor_arr);
}else{
    http_response_code( response_code: 404);
    echo json_encode(
        array("message" => "Nessun Dottore Trovato.")
    );
}
```

- Questo è il codice definito nell'endpoint chiamato. Fa da tramite con la funzione read() nella classe Dottori e la esegue.
- Sulla base dell'elaborazione eseguita crea un JSON da restituire al client chiamante insieme a un codice di stato http esplicativo dell'esito. Per tutte le altre operazioni (delete, update e create) sia di dottori che di altre risorse come le prenotazioni il procedimento è analogo. Ovviamente si è in grado di definire vincoli di ogni genere per l'ottenimento delle risorse private dell'azienda.

Centro Vista Android Application

Manuale d'uso

Centro Vista Android Application



1

The 'Sign in' screen of the Centro Vista Android application. It has a teal header with a back arrow and the text 'Sign in'. The form contains several input fields: 'Codice fiscale', 'Password' (with an eye icon for toggling visibility), 'Nome', 'Cognome', 'Email', 'Numero di telefono', and 'Data di nascita (YYYY/MM/DD)'. Below these fields is a dropdown menu for 'Sesso' with 'M' selected. At the bottom, there is a blue button labeled 'REGISTRATI'.

2

The 'Login' screen of the Centro Vista Android application. It has a teal header with a back arrow and the text 'Login'. At the top, there are two radio buttons: 'Dottore' (unselected) and 'Cliente' (selected). Below this, there are two input fields: 'Codice fiscale / ID Dottore' and 'Password' (with an eye icon for toggling visibility). At the bottom, there is a blue button labeled 'ENTER'.

Un utente non ancora acceduto al sistema può:

1. Registrarsi come cliente.
2. Eseguire Log In

Centro Vista Android Application

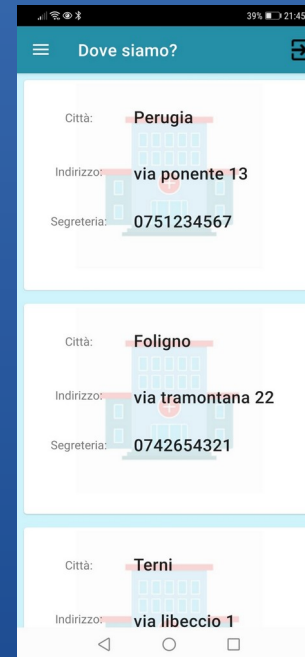
3



4



5



3. Consultare i dati relativi ai dottori.

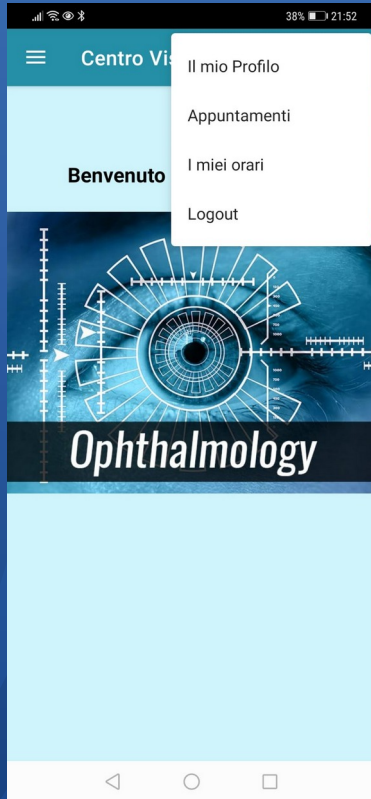
4. Orari di ricevimento

5. Consultare le informazioni relative alle sedi dell'azienda e le loro ubicazioni.

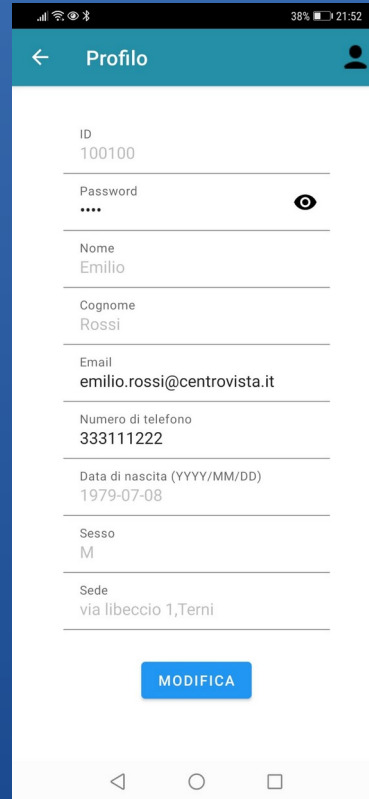
Centro Vista Android Application

Ruolo: Dottore

Centro Vista Android Application



1



2



Un dottore può:

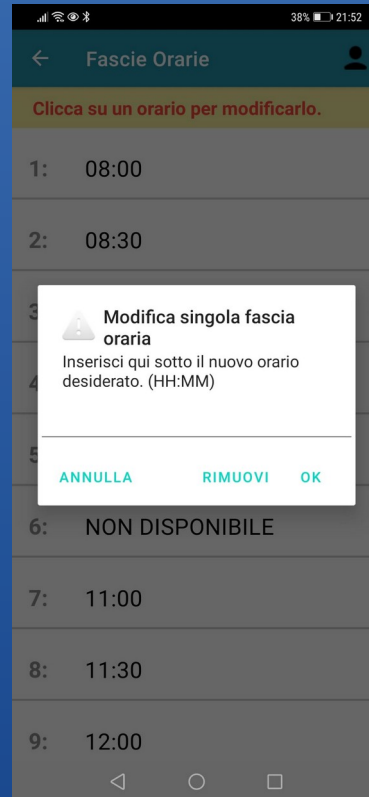
1. Accedere ai propri dati modificandoli se necessario.
2. Consultare i propri appuntamenti prenotati dai clienti.

Centro Vista Android Application

3



4



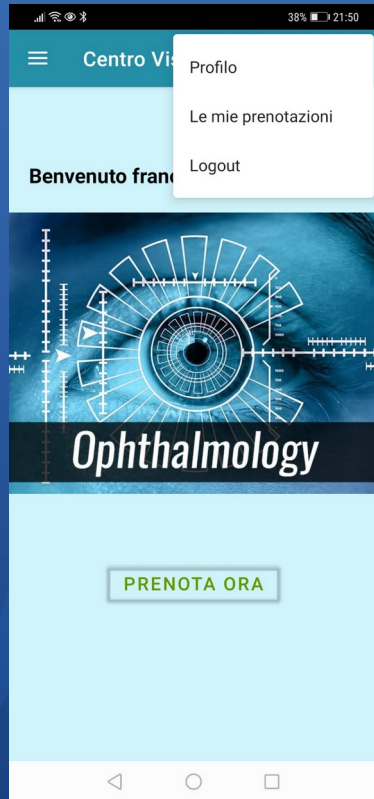
3. Accedere al pannello dei propri orari.

4. Modificare gli slot secondo le esigenze personali.

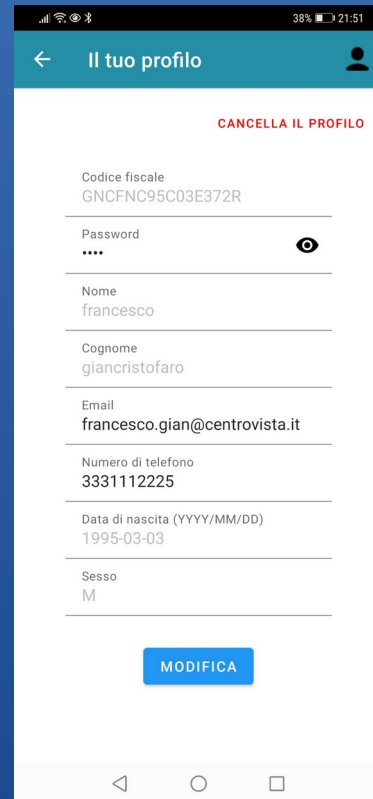
Centro Vista Web Application

Ruolo: Cliente

Centro Vista Android Application



1



2



Un utente
loggato può:

1. Accedere ai
suoi dati e
modificarli.

2. Prenotare una
visita

Centro Vista Android Application

3

Le mie prenotazioni

Clicca su un appuntamento per modificarne i campi.

ID: ccAq4GgP

Dottore: Emilio Rossi

Sede: via libeccio 1 Terni

Data: 2021-7-18

Orario: 08:30:00

Appuntamento creato correttamente.
Il codice è personale e unico. Usalo per modificare o cancellare la prenotazione in qualunque momento.

4

Le mie prenotazioni

Dottore:
Dr. Emilio Rossi [100100]

Sede:
via libeccio 1 Terni [3]

Data:

17	giu	2020
18	lug	2021
19	ago	2022

Fascia Oraria:

07	29
08	: 30
09	31

CANCELLA ANNULLA MODIFICA

3. Accedere alle proprie prenotazioni effettuate.

4. Modificare eventualmente una prenotazione secondo necessità (L'azienda dà molta flessibilità al cliente).

Parti di codice importanti

```
var branches: MutableList<Branch> = mutableListOf()
//http call to get the resources
val queue = Volley.newRequestQueue(activity)
val url = "http://centrooculistico.hostinggratis.it/centro_oculisticoREST/branch/read.php"
// Request a json response from the provided URL.

val jsonArrayRequest = JsonRequest(
    Request.Method.GET, url, JsonRequest<null>,
    { response ->
        for (i in 0 until response.length()) {
            val item = response.getJSONObject(i) as JSONObject
            val itemBranch = Branch(
                item.get("idBranch").toString(),
                item.get("address").toString(),
                item.get("city").toString(),
                item.get("switchboardNumber").toString(),
            )
            branches.add(itemBranch)
        }
        recyclerView = binding.recyclerViewBranches
        recyclerView.adapter = BranchItemAdapter(context, branches)
    },
    { error ->
        // TODO: Handle error
    }
)

// Add the request to the RequestQueue.
queue.add(jsonArrayRequest)
```

- Libreria Volley: vista inizialmente come possibile soluzione globale per il consumo dell'API, è stata infine utilizzata per tutte le richieste di tipo GET. I metodi da me studiati sono: `JsonArrayRequest()` e `JsonObjectRequest()`. Il limite di questi è che bisogna inviare e ricevere lo stesso tipo (o array o oggetti). La mia API invece per le richieste POST riceve in ingresso un oggetto JSON e restituisce un array JSON per cui non mi è stato possibile risolvere tramite volley.

Centro Vista Android Application

```
package com.example.centro_oculistico

import ...

object ServiceBuilder {
    private val client = OkHttpClient.Builder().build()

    private val retrofit = Retrofit.Builder()
        .baseUrl(baseUrl: "http://centrooculistico.hostinggratis.it/centro_oculisticoREST/") //
        .addConverterFactory(GsonConverterFactory.create())
        .client(client)
        .build()

    fun<T> buildService(service: Class<T>): T{
        return retrofit.create(service)
    }
}
```

- Libreria Retrofit: Si definisce un Servicebuilder.kt (nell'immagine a fianco), una classe contenente tutte le funzioni che interagiscono con l'API e un'interfaccia che setta gli header e il body delle varie richieste HTTP divise per funzione (slide successiva). Servono anche delle dataclass di supporto dove eseguire il parse del JSON che si riceve.

Centro Vista Android Application

```
interface Restapi {  
    @Headers( ...value: "Content-Type: application/json")  
    @POST( value: "user/read.php")  
    fun loginUser(@Body userData: UserLogin): Call<List<User>>  
  
    @Headers( ...value: "Content-Type: application/json")  
    @POST( value: "doctor/read.php")  
    fun loginDoctor(@Body doctorData: DoctorLogin) : Call<List<DoctorData>>  
  
    @Headers( ...value: "Content-Type: application/json")  
    @POST( value: "user/create.php")  
    fun createUser(@Body user : User) : Call<ResponseHTTP>  
  
    @Headers( ...value: "Content-Type: application/json")  
    @POST( value: "appointment/create.php")  
    fun createAppointment(@Body appointmentToCreate : AppointmentToCreate)
```

```
class RestApiService {  
    fun loginUser(userData: UserLogin, onResult: (List<User>?) -> Unit){  
        val retrofit = ServiceBuilder.buildService(Restapi::class.java)  
        retrofit.loginUser(userData).enqueue(  
            object : Callback<List<User>> {  
  
                override fun onFailure(call: Call<List<User>>, t: Throwable) {  
  
                    onResult(null)  
                }  
  
                override fun onResponse(call: Call<List<User>>, response: Response<List<User>>) {  
  
                    val user = response.body()!!  
                    onResult(user)  
                }  
            }  
        )  
    }  
  
    fun loginDoctor(doctorLogin: DoctorLogin, onResult: (List<DoctorData>?) -> Unit){  
        val retrofit = ServiceBuilder.buildService(Restapi::class.java)  
        retrofit.loginDoctor(doctorLogin).enqueue(  
            object : Callback<List<DoctorData>> {  
  
                override fun onFailure(call: Call<List<DoctorData>>, t: Throwable) {  
  
                    onResult(null)  
                }  
  
                override fun onResponse(call: Call<List<DoctorData>>, response: Response<List<DoctorData>>) {  
  
                    val doctor = response.body()!!  
                    onResult(doctor)  
                }  
            }  
        )  
    }  
}
```