## GPS

è un sistema di posizionamento e navigazione satellitare civile che, attraverso una rete dedicata di satelliti artificiali in orbita, fornisce ad un terminale mobile o ricevitore GPS informazioni sulle sue coordinate geografiche ed orario, in ogni condizione meteorologica, ovunque sulla Terra o nelle sue immediate vicinanze ove vi sia un contatto privo di ostacoli con almeno quattro satelliti del sistema. La localizzazione avviene tramite la trasmissione di un segnale radio da parte di ciascun satellite e l'elaborazione dei segnali ricevuti da parte del ricevitore.

## Firebase

Firebase è una piattaforma che fornisce numerosi servizi per lo sviluppo di applicazioni ad alto livello, tra i quali:

* Database realtime ad oggetti
* Archiviazione online di dati
* Autenticazione sicura
* Hosting di siti web (similmente ad Heroku o Altervista)

Riducendo i problemi legati all’infrastruttura dell’app e alla sua connessione con servizi remoti anche di diverso tipo (ad esempio AAA server, database e archiviazione limitata, limitazioni di costi e performance locando dei propri server a casa), si rende possibile una maggiore attenzione allo sviluppo dell’applicazione, in particolare alla sua interazione con i sistemi per i quali le applicazioni sono create.

Sono disponibili anche altre funzionalità accessorie, quali:

* Firebase Analytics: permette di ottentere un immediato riscontro sull’utilizzo dell’app e dei suoi problemi
* Test Lab: permette di testare le applicazioni Android su un largo numero di dispositivi virtuali contemporaneamente
* Notifiche: si possono mandare delle notifiche ai dispositivi che utilizzano l’applicazione
* AdMob: consente l’inserimento di pubblicità con la quale guadagnare denaro

**Database ad oggetti**

Sebbene i database relazionali siano dagli anni ’70 la scelta migliore per la memorizzazione permanente di grosse quantità di dati, negli ultimi anni, dato l’aumento continuo di necessità di informazioni immediate, si sono sviluppate nuove tecnologie NoSQL.

I database ad oggetti esistono sin dagli anni ’80, ma non sono mai stati presi in considerazione in quanto giudicati poco performanti.

Oggigiorno le motivazioni di una scelta di un OODB sono la semplicità nel design, un utilizzo più flessibile e, in alcuni casi, uguale velocità rispetto ai database relazionali.

La principale differenza con i database relazionali è la scelta di una maggiore disponibilità (percentuale di tempo di utilizzo produttivo del database) a discapito della consistenza; la maggior parte dell’elaborazione dei dati è, inoltre, delegata al software applicativo.

Il paradigma ad oggetti è assai diverso da quello relazionale, derivante dalla teoria degli insiemi, pertanto, molte operazioni, quali le join, non sono disponibili, tuttavia, le letture dei dati sono in media più veloci.

Un particolare sottoinsieme dei database ad oggetti sono i database a documento, dove per documento si intende una particolare codifica ed incapsulamento dei dati (ad esempio XML, JSON, GSON). I documenti possono essere intesi come oggetti in quanto non subiscono la rigida forma della struttura relazionale, in quanto non necessitano tutti le stesse sezioni, chiave, tipi di attributi.

Un programma, infatti ha molti tipi di oggetto, e gli stessi tipi, a loro volta hanno dei campi opzionali introducendo delle caratteristiche proprie. In ugual modo i documenti sono simili perché permettono di avere campi opzionali, sono facilmente ricodificabili e permettono la memorizzazione di molti oggetti in un unico punto di archiviazione.

Da finire

In a relational database, data is first categorized into a number of predefined types, and *tables* are created to hold individual entries, or *records*, of each type. The tables define the data within each record's *fields*, meaning that every record in the table has the same overall form. The administrator also defines the *relationships* between the tables, and selects certain fields that they believe will be most commonly used for searching and defines *indexes* on them. A key concept in the relational design is that any data that may be repeated is normally placed in its own table, and if these instances are related to each other, a column is selected to group them together, the *foreign key*. This design isknownas [*database normalization*](https://en.wikipedia.org/wiki/Database_normalization).[[2]](https://en.wikipedia.org/wiki/Document-oriented_database#cite_note-4)

For example, an address book application will generally need to store the contact name, an optional image, one or more phone numbers, one or more mailing addresses, and one or more email addresses. In a canonical relational database, tables would be created for each of these rows with predefined fields for each bit of data: the CONTACT table might include FIRST\_NAME, LAST\_NAME and IMAGE columns, while the PHONE\_NUMBER table might include COUNTRY\_CODE, AREA\_CODE, PHONE\_NUMBER and TYPE (home, work, etc.). The PHONE\_NUMBER table also contains a foreign key column, "CONTACT\_ID", which holds the unique ID number assigned to the contact when it was created. In order to recreate the original contact, the database engine uses the foreign keys to look for the related items across the group of tables and reconstruct the original data.

In contrast, in a document-oriented database there may be no internal structure that maps directly onto the concept of a table, and the fields and relationships generally don't exist as predefined concepts. Instead, all of the data for an object is placed in a single document, and stored in the database as a single entry. In the address book example, the document would contain the contact's name, image, and any contact info, all in a single record. That entry is accessed through its key, which allows the database to retrieve and return the document to the application. No additional work is needed to retrieve the related data; all of this is returned in a single object.

A key difference between the document-oriented and relational models is that the data formats are not predefined in the document case. In most cases, any sort of document can be stored in any database, and those documents can change in type and form at any time. If one wishes to add a COUNTRY\_FLAG to a CONTACT, this field can be added to new documents as they are inserted, this will have no effect on the database or the existing documents already stored. To aid retrieval of information from the database, document-oriented systems generally allow the administrator to provide *hints* to the database to look for certain types of information. These work in a similar fashion to indexes in the relational case. Most also offer the ability to add additional metadata outside of the content of the document itself, for instance, tagging entries as being part of an address book, which allows the programmer to retrieve related types of information, like "all the address book entries". This provides functionality similar to a table, but separates the concept (categories of data) from its physical implementation (tables).[[3]](https://en.wikipedia.org/wiki/Document-oriented_database#cite_note-5)

In the classic normalized relational model, objects in the database are represented as separate rows of data with no inherent structure beyond that given to them as they are retrieved. This leads to problems when trying to translate programming objects to and from their associated database rows, a problem known as [object-relational impedance mismatch](https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch).[[4]](https://en.wikipedia.org/wiki/Document-oriented_database#cite_note-6) Document stores more closely, or in some cases directly, map programming objects into the store. These are oftenmarketedusing the term [NoSQL](https://en.wikipedia.org/wiki/NoSQL" \o "NoSQL).

**Database Realtime**

## Inglese

The Car Finder

My exam project basically consists of an Android app that helps every forgetful person to remember where they parked their car.

My app is based on GPS functionality that nowdays is one of the most used service in the world. It is a [global navigation satellite system](https://en.wikipedia.org/wiki/Satellite_navigation) that provides [geolocation](https://en.wikipedia.org/wiki/Geolocation" \o "Geolocation) and time information to a [GPS receiver](https://en.wikipedia.org/wiki/GPS_receiver) like our android devices. The GPS system provides critical positioning capabilities to military, civil, and commercial users around the world. The United States government created the system, maintains it, and makes it freely accessible to anyone with a GPS receiver.

My app is made by a home where you can find a map,provided by Google, where the users can see the last car parking of their car and with a tap on the car parking marker,the app help the users by showing the route from their actual position to their car on the map.

In the homepage you can also find, on the top of the map, some buttons with their own functionality:

• a button that will list all users' carsparking saved in a database into another page

I have decided to use the google service of Firebase as a database that offers a effective object oriented database that in my opinion will be the future of databases. Firebase also offer an authentication system that it will make users able to log in to my app and save the car parkings in the cloud.

• a button that will make users able to edit their car parkings

• a button that will bring the users to their car showingthe route on the map,

Furthermorein the home page, on the bottom right corner, there is a round button with a blue "+" that will make users able to add a new car parking giving a name that will be saved in the database with the indication of time and the car parking position.

On the top left of the app there is a button with 3 lines that if clicked it will open a side menu where the users can find their name and email, if they are logged in ,and other functionality like :

• an option that will expand the map,with car parking markers and route, in fullscreen

• an option that will make users able to edit their cars parking

• an option that will show the info of the app for example the developers, the version and other legal stuffs