Adopted technologies

To offer the best decoupling and deployment flexibility we decided to follow the classic **MVC** web-application architectural pattern by developing a **REST Server** and a **Web Client**, while storing the data on a **NoSQL documental database**.
The excution flow follows the MVC paradigm: the data stored in the database is presented to the Web Client, from it it is possible to invoke functions on the REST Server that will then modify the aforementioned data, updating the Web Client view.

The reasons by which we decided to adopt a certain kind of tool or technology is explained in the following paragraphs.

Database

CouchBase Server NoSQL database: since the project aims to store a large and complex questionnaire for each patient adhering to the supervision programme of the melanoma cancer, we tried to find a database that was able to store a large amount of data whithout significantly changing the structure of the questionnaire itself. Couchbase is a documental NoSQL database whose datastructure is not composed by columns and rows as in the relational one, but documents are stored in the database in `.json` format, that is a simple, lightweight notation, compact and human readable. In Couchbase, a document usually represents a single instance of an object in the application code and each document can contain nested structures (sub-documents). This allows developers to naturally express many-to-many relationships without requiring a "reference table" or "junction table". So in this database we can store the entire questionnaire and all its sub-sections in an easy way and as naturally as possible. Morevover, the nested structure comes in handy when a certain degree of kinship is sought.
Morover, Couchbase offers the possibility distribute the storage: because of Couchbase’s architecture, an application can be developed at small scale, but also deployed to a distributed cluster, or separate clusters of varied topologies, without any architecture or behavioral changes to that application. This feature seemed to us to be very advantageous for a future distributed deployment of the application, as it emerged from the given documentation, this study was carried out in collaboration between three countries: Italy, Greece and Spain with the possibility of extension to other countries. In addition, each country has a network of centres where the collection of questionnaires is made, which could in turn be distributed in the country.
The last feature that made us decide to adopt this technology is its query lenguage: `N1QL` (pronounced “*nickel*”) is Couchbase’s next-generation query language. `N1QL` aims to meet the query needs of distributed document-oriented databases. It is very similar to `SQL`. `N1QL` gives application developers an expressive, powerful, and complete declarative language with industry standard ANSI joins for querying, transforming, and manipulating `.json` data – just like `SQL`.

REST Server

Spring: it is the standard-de-facto Java framework for what concerns the development of applications based on the MVC architectural pattern; it also comes with a generous amount of plugins that allow to easly connect to several types of database, deploy security filters, etc.
The following are the components that have been exploited:

- **Spring Boot**: TODO

- **Spring Data JPA for CouchBase:** Spring Data for Couchbase is part of the umbrella Spring Data project which aims to provide a familiar and consistent Spring-based programming model for new datastores while retaining store-specific features and capabilities. The Spring Data Couchbase project provides integration with the Couchbase Server database. Key functional areas of Spring Data Couchbase are a POJO centric model for interacting with Couchbase Buckets and easily writing a Repository style data access layer. It has been choosed to use this framework because of the many features provided (configuration support using Java based *Configuration* classes or an XML namespace for the Couchbase driver, *CouchbaseTemplate* helper class that increases productivity performing common Couchbase operations, annotation based mapping metadata but extensible to support other metadata formats, automatic implementation of *Repository* interfaces including support for custom finder methods...) and because already known and used before by the components of the group.
- **Spring Security and Json Web Token-based authentication**: TODO

Web Client

VueJS: it is yet another Javascript framework that recently exploded in popularity; it is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.
In a nutshell: Vue allows to bind the `.html` view to the data model, and to interactively modify the second by interacting with the first. Obviously, as explained in the following sections, Vue actually offers way more powerfull tools, such as a routing engine, local store, etc.