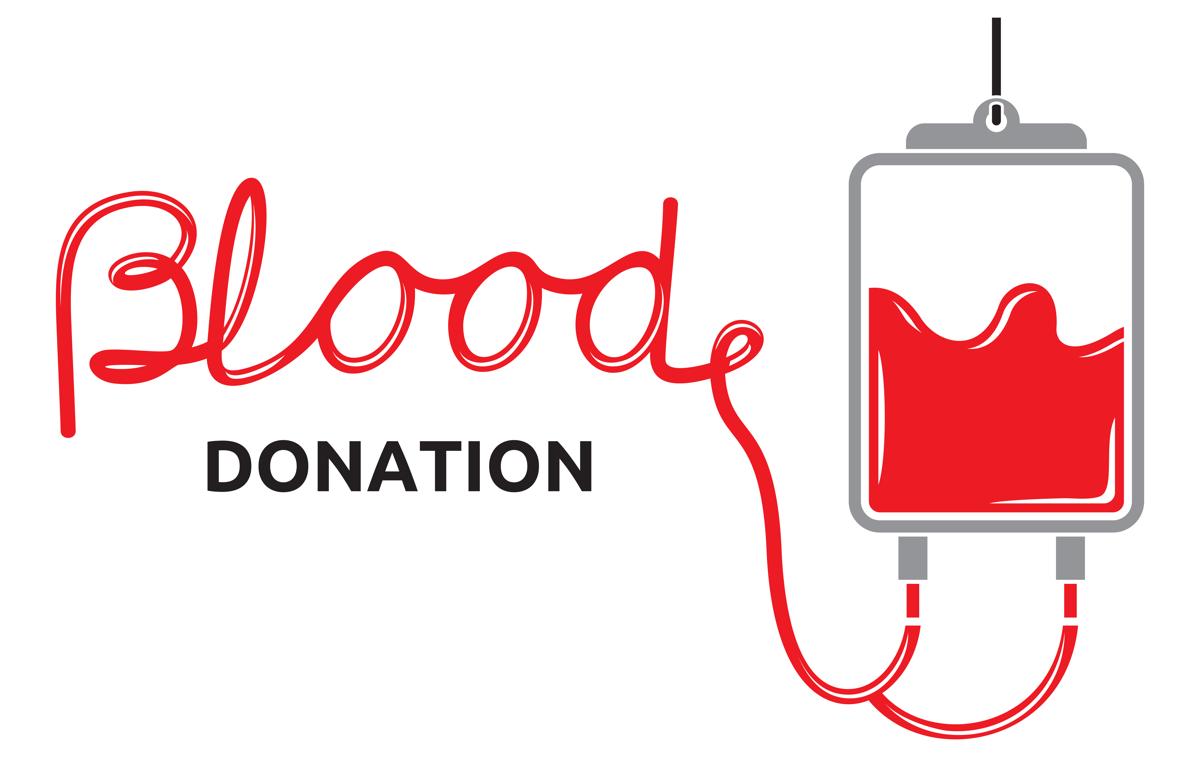
**Documentation**



Prof.Zsigmond Imre

Life iss good

Langa Valeriu – team leader

Ilisei Danut

Lazar Iulia

Lupascu Stefan

Manciu Daniel

Marusca Tabita

Maciuca Alexandra

Manaila Petrut

**Problem description**

* The application’s main focus is to encourage people to donate blood by offering an easy way to keep track of their donation details and analysis history.
* Blood donation eligibility requirements will be listed on the application.
* Persons who want to donate can register by filling a form with their personal data. After creating the account, they will be able to see their donation history and the results of their medical analysis. A donor can also choose to donate his blood to a specific person.
* Admins will provide and manage accounts for doctors so each doctor can request blood and then see the status of the request.
* In order to request blood, the doctor will have to fill a form about what blood they need.
* The staff (donation center personnel) processes all blood requests, manages the blood containers and contacts the donors in case there are not enough resources.
* The application provides a very flexible and accessible way of communication between doctors, transfusion centers and hospitals.

**Functionalities**

The user categories that we have in our application are **donor**, **doctor**, **donation center personnel (staff)** and **admin**.

**Donors** must be able to:

* See criteria for being able to donate.
* Register for blood donation by completing a form containing the following personal details: first name, last name, date of birth(day/month/year), address, address city/town, residence (only if the donor lives at another address than the previously specified one), residence city/town, country and of course a username and a password.
* View blood analysis history (i.e. result, date).
* View the next time they may donate (day/month/year).
* Change their personal details (i.e. any of the details specified during registration except the username).
* Specify the name of a patient, when donating to a specific person.

**Doctors** must be able to:

* Request blood by completing a form containing patient blood group, urgency level (high/medium/low), location.
* Check the status of the request (i.e. “sampling”, “in preparation”, “in biological quality control”, “redistributing”).
* See the available blood stocks of an area (county/city).

**Staff** (donation center personnel) must be able to:

* Collect donor data (i.e. from the form completed by donors).
* See hospital data.
* Manage journey of blood containers (i.e. change the status of the journey of containers).
* Process blood requests (i.e. determine priority, start a journey of requested blood containers).
* See available blood stocks of an area.
* See contact details of donors.

**Admins** must be able to:

* Add/Remove doctor accounts.
* Add/Remove hospitals.
* Provide/Revoke admin rights.

**Non-functional requirements**

**Security**

* The access permissions for system data may only be changed by the system’s data administrator.
* Passwords shall never be viewable at the point of entry or at any other time.

**Capacity**

* The application is able to support up to 200 users (this is the capacity of the free version of the database).
* 20MB (this is the capacity of the free version of the database).

**Performance**

* The connection between application and database takes 10 seconds.
* All functionalities must execute completely in less than 3 seconds.
* Any interface between a user and the automated system shall have a maximum response time of 3 seconds.

**Reliability**

* We use tried tested libraries with the most well known language at the core: Java.

**Portability**

* The application is desktop supported (Windows 10, Windows 8, Windows 7, Windows Server 2008 R2 SP1 64-bit, Mac OS 10.9+, Linux)

**Usability**

* The application is easy to use, intuitive, user-friendly.

**Project life cycle**

**The conceptualization phase**

In order to create this application, we firstly needed to understand better the problem by reading it many times so we figured which are the main story lines and flows.

We then had to decide which were the most suitable technologies for the solution of the problem and we agreed to use Java as the main programming language, PostgreSQL for the database, Github for version control, Trello for keeping track of our tasks and other tools that were helpful.

**The planning phase**

After making sure that we all understand what we have to do, we installed the necessary tools for developing the application and we got familiar with them.

The team lead also added constantly task on Trello, while the whole team decided to whom to assign them as fair as possible.

Diagrams were the next task we had to do. These are useful for us, because it helps us understanding the concepts and writing code. After we finished the class diagrams, we agreed that each of us works on least one diagram type, since we had to do use-case diagrams, class diagrams and sequence diagrams.

**The execution phase**

Since we finished working on diagrams, we got to the point where the coding part of the project is performed. Usually when developing a software application of such complexity implementation is the hardest part. However, we managed to do it pretty well and we did not have too much to worry about.

Our diagrams were so useful that the implementation has gone almost perfectly from the beginning. While a part of us worked on the back-end side, the others were developing the front-end part or testing, but we made sure to alternate these for each phase of the project.

We quickly fixed the bugs and warnings. We were lucky to have a team lead who reviewed our work at each step to make sure that everything is the right order.

After the code was developed, it was tested so that we are sure that all the requirements are functional.

We then started to work on the documentation and on the presentation.

**The termination phase**

When every part of the project is done, we did a re-evaluation of it. We took a closer look at everything and got to the conclusion that we achieved our goals.

**Technologies**

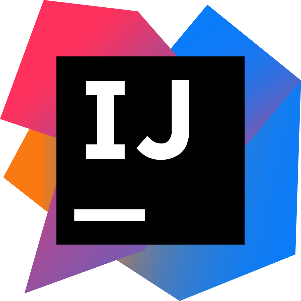
Since we decided to develop a web application, we chose the most suitable technologies for us.

**Java & Spring API & Gradle on Intellij IDEA**

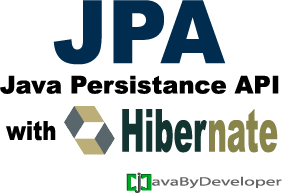
**Java** is a language we all are very used to working with, so we agreed that it would be easier for us to use it. It is a general purpose, [high-level programming language](https://www.webopedia.com/TERM/H/high_level_language.html) defined as an [object-oriented language](https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html) simplified to eliminate language features that cause common programming errors. Compiled Java code can run on most computers because Java interpreters and [runtime](https://www.webopedia.com/TERM/R/runtime.html) environments, known as Java Virtual Machines, exist for most [operating systems](https://www.webopedia.com/TERM/O/operating_system.html), including [UNIX](https://www.webopedia.com/TERM/U/UNIX.html), the [Macintosh](https://www.webopedia.com/TERM/M/Macintosh_computer.html) OS, and [Windows](https://www.webopedia.com/TERM/W/Windows.html).

**Spring Framework** is an [application framework](https://en.wikipedia.org/wiki/Application_framework) and [inversion of control](https://en.wikipedia.org/wiki/Inversion_of_control) [container](https://en.wikipedia.org/wiki/Servlet_container) for the [Java platform](https://en.wikipedia.org/wiki/Java_platform). The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java Enterprise Edition platform.

**Gradle** was designed for multi-project builds, which can grow to be quite large. It supports incremental builds by intelligently determining which parts of the build tree are up to date; any task dependent only on those parts does not need to be re-executed.

**Intellij IDEA Ultimate Edition** was a tool we all used before. It was JetBrains's first IDE. It is cross-platform and is primarily aimed at Java, Java EE and web development. An open-source version is available under the name IntelliJ IDEA Community Edition, and a proprietary version as IntelliJ IDEA Ultimate Edition.

**JPA Hibernate**

**JPA Hibernate** is an [object-relational mapping](https://en.wikipedia.org/wiki/Object-relational_mapping) tool for the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language. It provides a [framework](https://en.wikipedia.org/wiki/Software_framework) for mapping an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) domain model to a [relational database](https://en.wikipedia.org/wiki/Relational_database). Hibernate handles [object-relational impedance mismatch](https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch) problems by replacing direct, [persistent](https://en.wikipedia.org/wiki/Persistence_(computer_science)) database accesses with high-level object handling functions.

**JavaFX**

**JavaFX** is a [software platform](https://en.wikipedia.org/wiki/Computing_platform) for creating and delivering [desktop applications](https://en.wikipedia.org/wiki/Application_software), as well as rich Internet applications that can run across a wide variety of devices. JavaFX has support for [desktop computers](https://en.wikipedia.org/wiki/Desktop_computer) and [web browsers](https://en.wikipedia.org/wiki/Web_browser) on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://en.wikipedia.org/wiki/Linux), and [macOS](https://en.wikipedia.org/wiki/MacOS).

**StarUML**

**StarUML** is an open source software modeling tool that supports UML (Unified Modeling Language). It makes a clear conceptual distinction between models, views and diagrams.

**PostgreSQL**

**PostgreSQL** is an [object-relational database management system](https://en.wikipedia.org/wiki/Object-relational_database_management_system) (ORDBMS) with an emphasis on extensibility and standards compliance. As a database server, its primary functions are to store data securely and return that data in response to requests from other software applications.

**Github & SourceTree**

**Github** is a web-based [hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service) for [version control](https://en.wikipedia.org/wiki/Version_control) using [Git](https://en.wikipedia.org/wiki/Git). It is mostly used for [computer code](https://en.wikipedia.org/wiki/Source_code). It offers all of the [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of Git as well as adding its own features.

**SourceTree is** a free Git client for Windows and Mac. It provides a nice visual interface between user and Git. Managing branches in Sourcetree is easy. It enables the user to switch working copies with a single click. It also tells how far ahead or behind the user is to the version in the repository, and alerts to push or pull.

**Trello**

**Trello** is a [web-based](https://en.wikipedia.org/wiki/Web_application) [project management application](https://en.wikipedia.org/wiki/Project_management_software).  It’s a collaboration tool that gives you a visual overview of what is being worked on, who is working on it, and how far they’ve gotten.

The tool organizes your projects into boards, cards and lists.

**Tutorial**

**Tasks + percentages for each of us**