Project - Technologies

Presented by:

Daniel Libardo Diaz Gonzalez - *dandiazgo@unal.edu.co*
Andrés Felipe León Sánchez - *anleonsa@unal.edu.co*
Alejandro Medina Rojas - *alemedinaro@unal.edu.co*
Angel David Ruiz Barbosa - *aruizba@unal.edu.co*

Professor:
Oscar Eduaro Alvarez Rodriguez
oalvarezr@unal.edu.co

Friday 11th of July

Universidad Nacional de Colombia
Facultad de Ingeniería
Ingeniería de Sistemas y Computación
2025

CONTENT

1. Technologies implemented:

Due to the nature of the Kodetron project, which consist in a code editor for competitive programming, we decided to look for a technology with a low cost in terms of resources and a high and effective performance at the moment of execute them in a desktop environment as the one specified in the project rules, at the same time we needed a technology with a relevant number of frameworks and tools that could allow us to develop all the parts of the application in a more effective way.

In the next table we are going to expose the main advantages and disadvantages of these technologies and the main reasons why we chose them over the other alternatives.

Front-end technologies:

Technology	Advantages	Disadvantages
C++(Qt version 17)	Native performance, rich UI components, cross-platform. It also has a robust tool called Qt creator which allows a more effective way of creating the UI	If it isn't used commonly by the developer, the learning curve can be high at the beginning
Java (with JavaFX)	Mature, portable, and good documentation.Rich UI libraries.	Runs on JVM, which can affect performance.Slower than native apps.
Python (PyQt/PySide)	Has a simpler syntax	Performance issues for complex applications.Not compiled.
Electron (JS)	Modern frontend framework inspired in Javascript, easy for web development.	Very high resource usage.Poor performance for heavy tasks.

As we can see, C++ was the most optimal and effective option for the purpose and nature of this project, due to its high performance in desktop applications such as the one of the project also allowing us to run the editor in windows, macOS and linux. At the same time, it presents a reliable and easy to use framework, which still does not generate major changes on the performance of the application.

By the other hand, the other technologies if well can also generates an effective UI and could have a more simple syntax, their high resource usage and the lack of frameworks for desktop applications, could have generated performance issues which are crucial at the moment of developing a code editor such as Kodetron

Back-end technologies:

Technology	Advantages	Disadvantages
C++(Crow version 17)	Lightweight and fast.Modern C++17 syntax.Easy REST API definition.No external dependencies.	Smaller community of developers.
Javascript(Node.js)	Web-based, large ecosystem, and it has a large community	Heavy RAM usage, poor native performance due to its web focus
Java(Spring Boot)	Mature, good tooling, standard in enterprise environments..	Requires JVM, verbose syntax, slower runtime.
Fast API (Python)	Extremely fast.Intuitive syntax.Great for prototyping.	Python backend is more optimal for large amount of data, its running time is slow in native applications

As we can see on the table, the alternatives presented by other programming languages and frameworks if well can have a larger community of developers or a simpler and more intuitive syntax, their performance at the moment of executing the application on a desktop environment such as the specified for the project presents a huge flaw in comparison with the C++ framework, which is crucial at the moment of handle tasks with a high performance requisite such as editing and executing code.

It is also important to note that C++ because it is the closest programming language on the table to the memory and hardware of the computer, is a more reliable and effective option for the project, because in this one we are constantly editing and executing files inside the hardware of the computer.

2. Data base;

For the Kodetron project we were looking for a database technology with: a low resource consumption and an easy set of instructions to interact and modify the data, also it is important to point out that due to the low amount of data that our project is gonna handle at this stage, we decided to prioritize speed over the administration of a high number of data.

First is important to point out that we used a SQl database system also known as a relational database, this is not just because of the rules specified for the project, but also because as we are not dealing with a huge amount of data or with a structure that is modifying itself constantly during the time, we can rely on a SQL data base system due to its simplicity and performance for native applications such as Kodetron

Because of this, in the next table is gonna be represented the different data base ecosystems and the advantages and disadvantages of each one of them:

Technology	Advantages	Disadvantages
SQLite (Selected)	Lightweight and serverless.File-based, easy to deploy.Fast for small to medium data.Fully ACID-compliant.	Not designed for high concurrency.Not suitable for large-scale apps.
MySQL	Widely used.Good for multi-user and web apps.Broad ecosystem.	Requires server setup.Overhead for small local apps.
PostgreSQL	Advanced features.Highly reliable.Good performance with large datasets.	Requires more configuration.Heavier footprint.
MariaDB	Fork of MySQL.Good performance.Open-source focused.	Similar drawbacks to MySQL.Still server-dependent.

As we can tell from the table and the description previously given, the technology that is more suitable to our needs is SQLite, this is because it is fast , effective and as our project do not have a huge amount of data concurrency as it could happen on a enterprise app or a social media, we don't need a good performance with a large dataset or a huge ecosystem, but just a simple, fast and reliable data base which the user would not have too much interaction with besides a simple amount of instructions.

Finally an important aspect to take into account in justification of the use of SQLite, is its efficiency at the moment of the implementation with the different frameworks such as Qt and Crow, due to the simplicity and scalability of the SQLite distribution.

3. Tools and libraries;

For the development of Kodetron we also decided to use a new set of AI based tools that have been gaining relevance over the recent years due to their effectiveness at the moment of reducing the amount of time spent writing code and executing test cases by the developer.

The tools specifically used by us for the project are the coding assistant Github copilot and the AI based code editor called Cursor, these tools are crucial at the moment of writing large amounts of code lines and writing a substantial number of test cases that verifies the efficiency of the application. Allowing us to implement more time designing the software structure and design, generating a more effective and efficient application.

It is also important to point out that we are always checking the code generated by the AI tools, this is because even in a stage when they can generate simple applications with a single prompt, they can usually misunderstand important aspects of the project or even generate inconsistencies on the syntax of the code. And also, we decided not to rely completely on AI, this is because besides we desire a smaller development time, we also want to keep an effective and centralise control over our application.