LLVM IR Translator

0. Authors

Vladimir Abramenko

1. Introduction

Web client-server application that allows users to download code in different languages (C, C++, Python, Java and C#) and receive its equivalent in LLVM IR format.

2. Glossary

llvm IR- is a low-level programming language used as an intermediate step in the compilation process.

llvm - project with a collection of modular and reusable compiler and toolchain technologies.

3. Actors

User. He wants to translate his code into ir format

4. Functional requirements

4.1.1. Use-case <IR format>

Actors: User.

Goals: See the ir format of desirable code.

Precondition: The user needs to have his own code, which he can upload to our server.

Trigger condition: The user wants to see the ll file of code.

Mains success scenario:

1) User loads code

2) Code appears

3) Text of IR appears

4.1.2. Use-case <Saving files>

Actors: User.

Goals: Saving files into a database to see them in future or work with them.

Precondition: The user needs to be registered.

Trigger condition: The user wants to save files with flags of optimizations.

Mains success scenario:

1) User loads code

2) Code appears

3) Text of IR appears

4) Saving both files

4.1.3. Use-case <Applying of optimization>

Actors: User.

Goals: Optimize code with specific flags to see differences.

Precondition: The user needs to have his own code, which he can upload to our server.

Trigger condition: The user wants to optimize IR.

Mains success scenario:

1) User loads code

2) Code appears

3) Apply optimization

4) Text of IR appears

4.1.4. Use-case <Load files>

Actors: User.

Goals: Load files from db.

Precondition: The user needs to be registered.

Trigger condition: The user wants to get his files.

Mains success scenario:

1) User choose which files he wants to get

2) Code appears

3) Text of IR appears

5. Non-functional requirements

5.1 Performance

Request processing time should not exceed 5 seconds for a code of average complexity.
The system must support simultaneous requests from at least 100 users.

5.2 Safety

Protection against SQL injections and XSS attacks.
Encryption of user data.
Limit the size of the downloaded code.

5.3 Scalability

Ability to add new programming languages in the future without significant architectural changes.

6. Development stages

1. System architecture design.
2. User interface development.
3. Implementation of the server part and API.
4. User interface realization.
5. Integration of translators for each language.
6. Testing and debugging.
7. Development of documentation and reference materials.
8. Starting and monitoring the system.

7. Expected results

1. A working website that allows users to translate code in specified languages into LLVM IR.
2. Documentation on the use and support of the system.