A black and white logo

Description automatically generated with medium confidence**Софийски университет „Св. Климент Охридски“**

**Факултет по математика и информатика**

**КУРСОВ ПРОЕКТ**

**по**

**„Elixir“**

**Изготвил:**

Мартин Живков (ФН: 72071)

„Информационни системи“ 3 курс

* Въведение:

Проектът ми е на тема Peer-to-peer file sharing, което представлява нещо наподобяващо Торент. Разбира се, в много опростен вариант. Това е система за обмен на файлове, която използва централен сървър за откриване на потребители и данни.

* Системата:

Системата се състои от две части:

1. Сървър, който съхранява метаданни за наличните файлове.
2. Клиенти, които теглят файлове от други клиенти – клиентите взимат информация от сървъра за това къде (от кой клиент) даден файл може да бъде свален.

Системата предоставя функционалност за сваляне на файлове от различни потребители.

Всеки потребител може да сваля файл от всеки друг. Има централен сървър, който съхранява информация за потребителите и файловете, които могат да бъдат свалени от тях (файлове не се свалят и съхраняват на централния сървър).

Сървърът съхранява информация за активните потребители в паметта – имената, адресите и портовете им (p2p обмен изисква да се знаят адресите на peer-ите); файловете, които могат да бъдат изтеглени от тях (абсолютен път). Тази информация се обновява, когато клиент регистрира файлове за изтегляне и когато клиент затвори връзката със сървъра.

Клиентите могат да се свързват един с друг За целта се реализира „мини сървър“. Този мини-сървър може да обработва командата download,

* Клиент

Клиентът изпълнява следните команди:

1. register <user> <file1, file2, file3, …., fileN> - позволява на клиентите да „обявят“ кои файлове са налични за сваляне от тях. Чрез параметъра username, потребителят може да зададе свое уникално име (името, с което сървърът ще асоциира съответното IP).
2. unregister <user> <file1, file2, file3, …., fileN> - потребителят обявява, че от него вече не могат да се свалят файловете <file1, file2, file3, …., fileN>.
3. list-files – връща файловете, налични за изтегляне, и потребителите, от които могат да бъдат изтеглени.
4. download <user> <path to file on user> <path to save> - изтегля дадения файл от съответния потребител.

* Сървър

Сървърът може да обработва изброените команди по подходящ начин:

1. При получаване на register <user> <file1, file2, file3, …., fileN>, сървърът обновява информацията за този потребител (добавя информация, че тези файлове са налични за сваляне от съответния потребител).
2. При получаване на unregister <user> <file1, file2, file3, …., fileN>, обновява информацията за този потребител (съответните файлове вече не са налични за сваляне от този потребител).
3. При получаване на list-files, връща регистрираните в сървъра файлове във формат <user> : <path to file>
4. При изпълняването на download <user> <path to file on user> <path to save>, не се случва комуникация с централния сървър.

**a.** Клиентът определя IP адреса и порта на потребителя, от който може да бъде изтеглен даденият файл

**b.** Клиентът изпраща командата на мини-сървъра на потребителя, определен в стъпка (а)

**c.** Мини-сървърът изпраща файла

**d.** След като потребителят получи файла, автоматично изпълнява командата register . По този начин информацията в главния сървър за потребителите, притежаващи този файл, се обновява.

* Реализация

За централния сървър съм използвал GenServer, за да пази състоянието на метаданните и за по-структуриран код. За комуникацията между клиентите съм използва вградената библиотека на Erlang :gen\_tcp за стабилна връзка. За самото сваляне на файлове съм решил да използвам Task, понеже се извършват side effect действия.

* Бъдеща оптимизация

Да работи правилно програмата. Да се направят подобаващи съобщения за грешки. Вместо tcp протокол да се приложат NIF-ове (отворената библиотека QUIC, написана на C).