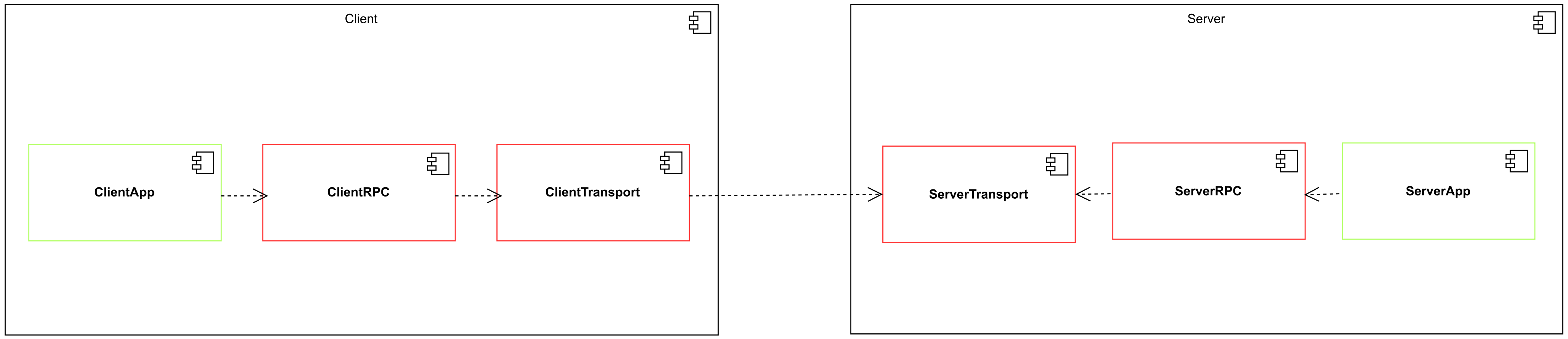

Imagine you are tasked with drafting the architecture for a proprietary RPC framework.
The framework shall provide easy remote function calls.
The component-diagram above is the rough requirement how the overall architecture shall look like.
Consider the Client-App and Server-App as clients of your framework.
The transport protocol shall be HTTP but this is not yet finalized.
For core functionality of RPC and HTTP, 3rd party libraries shall be chosen in future.
Use placeholder names "Dream-RPC" and "BadDream-HTTP" for them, and try to anticipate and fake the core features.
For the remote functions, only primitive parameter and return types are required --> Complex object transfer is not required!

Your focus should be on designing the foundational functionality of the RPC framework, emphasizing straightforward remote function calls and basic data transfer.
While HTTP offers a broad range of capabilities, and RPC frameworks can be highly complex, your goal is to sketch a clear, simple, and effective architecture.
Avoid delving into advanced features such as encryption, or caching at this stage.
Aim for a design that demonstrates how the core components interact and handle communication efficiently

1. Draw your vision of an architecture!
    1. Format png, jpg or svg.
2. Create a class-diagram which describes the interfaces of ClientRPC and ClientTransport and the relations between them.
    1. The paradigm must be "Object Oriented Programming".
    2. Format png, jpg or svg.
3. Implement the aggregation of the interfaces you have drafted in point 2.
    1. It shall be a command line executable without UI.
    2. Choose an object oriented programming language out of these: C++, C#, Java, Python.
    3. Fake the concrete implementation of the 3rd Party libs.
    4. Fake the whole server.
    5. The program shall...
        1. ... open a connection to the fake-server.
        2. ... call the remote functions 'string hello() { return "Greetings!" } ' and 'int add(int a, int b) { return a + b }', the calls, parameters and return values must be printed to command line.
    6. The fakes shall print out information to command line, so the calling order can be comprehended.
    7. Provide an build script and all dependencies. (Common dependencies like java runtime or ms-redistributable as link)
    8. Put everything in a GitHub repository

The Time Window for the tasks shall not be more than 3 to 5 hours. This shall give you orientation how extensive the solution shall be.
Keep it simple, and stay on the main road!

Keep the source files of the figures, perhaps we will ask you for changes, live in the next interview.