# SignalR – Context view

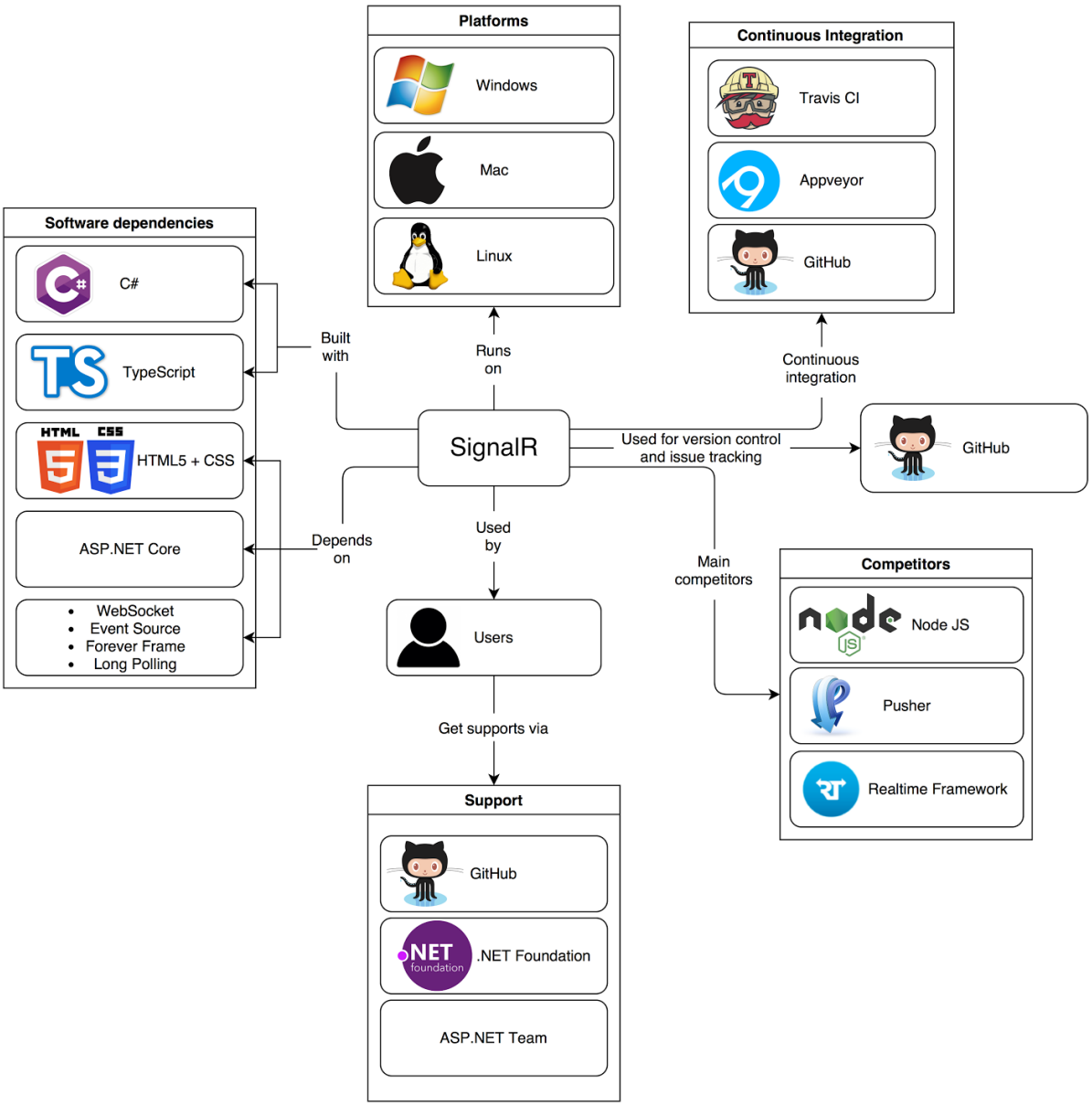
Nicolas Ragnell

39168

Nicolas.ragnell@abo.fi

# Context view

The context view describes a system's relationships, dependencies and different kinds of interactions between the system and its users. It describes how the system interacts with the outside world. The diagram below describes all the dependencies and relationships.



## Software dependencies

The SignalR library is written in C# and the client-side uses TypeScript. The library depends on ASP.NET Core, since it is an extension to it and extends its functionality. SignalR adds real-time functionality to developer's applications.

C# is an object-oriented language developed by Microsoft that you can use to create client-server applications, Windows client applications [1] and much more, which makes C# a good candidate for SignalR. TypeScript is open source, and is also developed and maintained by Microsoft. JavaScript users can easily use TypeScript, since it uses the same semantics and syntax. TypeScript also compiles down to JavaScript [2].

SignalR implements several transports, and automatically selects the best available transport for the given case. These technologies are WebSockets, Event Source, Forever Frame and long polling. SignalR mainly uses WebSockets when it's available, and if it isn't, it tries Event Source, and if that isn't working, it goes with Forever Frame. The last option is the real-time-emulating technology called long polling.

In the best-case scenario, only WebSockets is used. In that case, SignalR provides a fully bi-directional [3] communications protocol. WebSockets is supported by most major browsers, since the requirement for usage is HTML5. However, there are servers who do not support it.

Event Source is the plan B, and it is supported by [4] all major browsers except Internet Explorer and Microsoft Edge. It is similar to WebSockets, except that EventSource only allows [5] half-duplex communication. It is a more lightweight solution in cases where fully bi-directional communication is not needed.

Forever Frame [6] is the plan C for SignalR connections. The concept is based on chunked encoding, where data is divided into blocks. The scripts are executed in the order they are loaded to the HTML page. Two major setbacks with Forever Frame are the lack of error handling and the lack of states during the loading process.

With long polling [7] the server holds a client "request open until new data is available". When the client receives the data, it sends a new request and continues polling the server. Questions about the backend arise when a system with long polling

has a growing user base, as there are thousands of clients continuously polling the server which is most of the time totally unnecessary. Long polling emulates real-time, but is another variation of polling.

## Main competitors

There are several big competitors when it comes to real-time web applications, with the main competitors being Node.js, Realtime Framework and Pusher. Comparing SignalR to the others might be a bit difficult since SignalR is still in alpha release, but you can still find some differences as well as similarities between them.

Pusher, stating that they are the “leader in real-time technologies”, has a 53% market share in Real-time backend technology [8]. They implement fallback mechanisms [9], similarly to SignalR, granting real-time connectivity in almost any given case. Pusher is free to use to a certain extent (100 max connections), but requires that you to choose a paid plan if you need more connections, unlike SignalR that is completely free to use.

RealTime Framework is quite similar to the other competitors, but also requires you to pay if you plan to use their services on a larger scale. They offer a free option which is limited to 100 simultaneous connections. They are also smaller than Pusher, with a 14% market share. Realtime focuses on mobile devices [10] and messaging applications.

Node.js is open source, free to use and is designed to build expandable network applications. Depending on what you are going to develop, both Node.js and SignalR are viable choices. For example, if you are working with .NET systems SignalR is the obvious choice. On the other hand, Node.js has been on the market for a long time (since 2009) and there is a lot more documentation and support when using it. Now, Node.js has a 6% market share in the web framework category [11].

# References


1. https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework
2. https://www.typescriptlang.org/
3. http://blog.teamtreehouse.com/an-introduction-to-websockets
4. http://caniuse.com/#feat=eventsource
5. https://www.sitepoint.com/implementing-push-technology-using-server-sent-events/
6. http://www.webreference.com/programming/javascript/rg30/index.html
7. https://www.pubnub.com/blog/2014-12-01-http-long-polling/
8. https://siftery.com/pusher
9. https://pusher.com/docs/fallbacks
10. https://framework.realtime.co/messaging/
11. https://siftery.com/nodejs