Deployment considerations

The focus of this project is on building decentralized applications on top of existing application platforms. It is not the intention to design and build core Blockchain protocols from the ground-up, but instead focus on developing software on top of this fundament.  
Blockchain development platforms (and deployments in general) come in different variants. Either you deploy in public or in a consortium-private\* context. The prominent Blockchain application platforms available are Ethereum and Hyplerledger Fabric.

**Ethereum** is a public, open source (GPL) Blockchain application platform, allowing for the deployment of smart contracts which can be interacted with via native wallets and web-based applications and plugins. The advantage of using Ethereum is that the infrastructure requirements are covered, meaning the supporting P2P network is available for direct use, offering both a test and live/production network. Additionally, Ethereum benefits from a vibrant online community and a multiplicity of development tools which are freely available. Therefore, Ethereum is a good starting point for understanding Blockchain basics, allowing for a steep learning-curve due to network effects (contract code itself is ‘public’ as well). Downside to public deployments are transaction costs, and fixed capacity constraints in smart contract processing.  
Smart contract related code is executed via the Ethereum Virtual Machine (EVM) running on network nodes, at a limited price (i.e. ‘gas’) which is paid for in ‘Wei’, the smallest unit of measure for the Ether cryptocurrency.   
Microsoft is sponsoring Ethereum.

**Hyperledger** **Fabric** is a consortium, open source (Apache 2.0) Blockchain application platform, allowing for the deployment of chaincode, custom implementation of authorization concepts (certification) and consists of a modular architecture allowing for pluggable components (e.g.: consensus, validation protocols). Considering that the currently available consensus protocols are foremost based on PBFT, this application platforms is predominantly oriented towards small-sized network deployments (i.e. low node scalability).   
Since that Fabric is private, no infrastructure is readily available. This means that nodes must be setup by yourself. Guides are available for this purpose. IBM also offers production-ready Blockchain services via Bluemix, providing a Blockchain network of nodes for test purposes.   
Chaincode is executed in virtual Docker containers via Docker images created for a given operating system running on a Blockchain node. Since it is private, Fabric does not feature a native cryptocurrency nor does it expose hard restrictions on operations. A consideration for private deployments is that, unlike public deployment, anonymity is not feasible since some form of administration is required to participate. Currently, performance (tps, latency) in consortium chains is far better than public, due to its expensive PoW consensus algorithm.   
IBM and Intel are sponsoring the Hyperledger foundation and its projects.

*\*Although completely privately owned, intra-organizational Blockchain deployments are theoretically conceivable, we do not consider this a practical deployment scenario which appreciates the key Blockchain capabilities. Corsortium deployment refers to an interacting network of participating entities. Therefore any reference to private applications in this document refers to consortium deployments.*

Dapp Development tooling and info:

Ethereum:

To develop Dapps on Ethereum, different tools and networks are available. Note that what is described here is limited to the tools used during this project.

**Ethereum Studio** is a web-based IDE provided by Cloud9 (Microsoft Azure) and hosted by ether.camp. If offers a virtual Linux Ubuntu workspace for developing complete Dapps. It features a sandbox environment for testing smart contracts, a local HTTP server for testing web-based applications, a Solidity debugger, an EVM compiler, network integration via RPC URL to directly deploy contracts to either test or live networks and a CLI supporting git integration. Because it is free and comprehensive, Ethereum Studio was our initial development environment of choice.   
Signup to ether.camp is required, [here](https://live.ether.camp/).  
After starting the sandbox, you can click ‘contract’ to determine the contract ABI. The ABI should be used for the application and must be updated every time variables are declared ‘public’ or when a function is adapted. The local http server can be found at <https://[username].by.ether.camp:8080/>.

**Remix** is a web-based IDE with more limited functionality than Ethereum Studio, but has the advantage of integrating with the Metamask Plugin and by directly interacting with smart contracts on test or live networks.   
For example, when specifying a contract address, you can call ‘public’ variables to have their value returned from the IDE itself. When performing a function on a contract, Metamask notifications are automatically triggered.   
Remix was used when Ethereum Studio could no longer be used for sending contracts to the test network.

**Mist Wallet** and **Ethereum Wallet** is software for connecting to the Ethereum networks, manage accounts, transmitting Ether via public keys, creating contracts and calling functions in contracts. The Mist Wallet additionally features a browser supporting web3 services, allowing for online Ether payments for web applications enabled for web3. This is an alternative to the Metamask plugin. The remix IDE is embedded into the Mist wallet.

**Metamask** is a freely available chrome plugin that injects web3 services into the browser, allowing for transactions to be signed from Chrome. Whenever a web3 transaction is caught, a Metamask notification is prompted requiring your approval.

**Etherscan.io** is a website for tracking transactions, contracts and their statuses on the differently available networks. It provides a log of transactions per contract. **live.ether.camp** allows for the same, but limited to the main network and proves less reliable. Pro for live.ether.camp is that you can request free Ether, which will then send 5 Ether to your account so that this may be used in testing.

**https://www.4byte.directory/signatures/** is a useful website for determining contract function ID’s and data types. When invoking contract functions, a function ID should be specified (e.g.: 0xdce293a7) along with the payload.

**Test networks (RPC URL):**   
https://ropsten.infura.io/  
https://ropsten.infura.io/metamask

General**:**

**Github** is a collaborative, distributed version control system for projects. Advantage of using Github in combination with Ethereum Studio is that you can perform pull and push requests form the bash CLI. The Github repository used for this project can be found [here](https://github.com/ExperisCiber/CiberBlockchain).