FOSS IoT Frameworks review

-

OCCIware Project

Benoit RENAULT

2017/08/10

Contents

Introduction

The context for this document is the need for an IoT middleware platform for building, managing, and integrating connected products for the OCCIware LinkedData Use Case. As the OCCIware project is FOSS[1], the component **must** be open-source itself. Therefore, what you will find here is a short state of the art of such frameworks, and no consideration shall be given to proprietary solutions.

Please note that, while the evaluation criteria themselves may be considered timeless, the analyses presented below only reflect the state of the different projects as of this document's last update date, written on the first page.

The criteria are mainly chosen so that each solution needs not be analyzed more than three hours, and only by consulting the website, the documentation and the project's repository, never actually executing the project.

If you spot a mistake or confusion within the document, please don't hesitate to either open a Github issue on the source repository or directly fork it, modify your clone, and make a pull-request for the author to easily merge your modifications if deemed appropriate.

License

[1] Free or Open-Source Software

1 Evaluation criteria

Each of the criteria is presented below with at least one argument as to how it allows to measure a given characteristic. **For recall, the criteria are also chosen so that each solution needs not be analyzed more than three hours, and only by consulting the website, the documentation and the project's repository, never actually executing the project.** For qualitative criteria, a little argumentation will be given in the detailed analysis (noted with excellent/good/average/poor/non-existent).

Some of them have been inspired by an O'Reilly report on IoT platforms, and the security-oriented ones by Craig SMITH's "IoT Framework Assessment" OWASP page.

1.1 General criteria

Here are some general criteria that allow us to assess the quality of any FOSS project.

Project Health Determines how "mature" the project is, and its potential to last on the long term.

- First Release Date: The older, the better, might clue a position of pionneer, or a certain stability.
- Latest Release Date: The newer, the better, might clue that the project's release cycle is still active.
- Latest Commit Date: The newer, the better, if the project's release cycle is slow, might clue that the project is still ongoing active development.
- Number of main contributors: The higher, the better. If equal to one, exercise caution. Main contributors have proportional contributions to the code repository.
- Open issues ratio: The lower, the better. Is equal to the number of open issues over the total number of issues. Might clue at a good responsivity to user input, be it for bugs or improvements.

Company backing Partly shows how reliable the software can be: if there is a company ready to back it up and provide support to customers, and if there are other companies using the software, it might mean that it is trustworthy.

- Company Support: The better the company support is, the better the appreciation.
- Company Adoption: The more the software is adopted by other actors, the better the appreciation.

Documentation A well-documented project is a must.

- Currentness: The documentation must be up to date with the code. The more it respects this principle, the better the appreciation.
- Adaptability: Documentation must be adapted to the different users. The more it respects this principle, the better the appreciation.

UI-UX The software must provide appropriate tools at all levels, be it CLIs or Graphical UIs, depending on the context.

- Server Management UI: the more powerful and concise, the better the appreciation.
- Sample applications: the more there are and the more complete, the better the appreciation.

1.2 IoT-specific criteria

Here are some general criteria that allow us to assess the quality of IoT projects in particular.

Security A key characteristic for IoT systems. As we cannot conduct a security assessment ourselves, we will rather look for a commitment to security and whether audits have been made or not. Security must be ensured at all levels: connectivity, storage, update, authentication, appropriate management of devices...

- Statements: the more is explained on the security measures taken in the documentation, the better the appreciation. Clues at a concern and understanding of the IoT problematics.
- Audits: the more positive reviews about the security are, the better the appreciation. Clues at a good applications of best security practices.

Connectivity/Flexibility Is very important for IoT systems, because they are supposed to connect many elements that are all very different.

- Number of compatible hardware platforms: The higher, the better. Might clue to a shorter time-to-market.
- Number of supported protocols: The higher, the better. Might clue at a will not to lock the user in the system.
- Number of SDK implementations: The higher, the better. Might clue at an easier time for in-house developpers to incorporate the solution with their own prefered language.
- Modularity: the more the components of the solution can easily be changed (for instance, the DB), the better the appreciation.

Scalability As IoT systems tend to grow really fast, it is necessary that the application scales by design, and allows for many devices, all over the world, to connect to it, without failure.

- Third-Party scalability: IoT solutions are often a combination of many pre-existing FOSS projects. This criteria evaluates whether the 3rd-Party solutions chosen by the IoT platform are capable of scaling: the more they are, the better the appreciation.
- Platform scalability: IoT solutions must also be scalable themselves by being capable of being distributed across different machines and still be able to talk to one another smoothly. This criteria evaluates whether the platform is capable of scaling: the more it id, the better the appreciation.

1.3 Application-specific criteria

Here are some specific criteria that are related to the peculiar needs of the OCCIware project.

Arduino/NodeMCU compatibility Since it is the most popular platform for hardware experimentation, it is an important criteria that the service musts provide good support for it.

- Library: The more complete and recognized the library, the better the appreciation.
- Boilerplate code: The smaller the boilerplate code, the better the appreciation.

Java The framework must be based on Java for easy editing of the sources and better maintainability, since it is the main language of the author.

- Server percentage: The higher the quantity of Java code for the server, the better.

2 Detailed analysis

2.1 SiteWhere

Project Identity

Corporate URL	http://www.sitewhere.org/
Documentation URL	http://documentation.sitewhere.io/
Repository URL	https://github.com/sitewhere/sitewhere
License	CPAL-1.0
Main Selling Point	Highly flexible

Project Health

- First Release Date: The development started in 2010 as a closed source, asset tracking platform. The first open source release happened on **2014/03/06**. With 7 years of age, it can be assumed it is quite mature and stable.
- Latest Release Date: **2017/06/19** - Still releases new versions as of today. The project has an irregular release cycle, "When it's ready"-Debian-like.
- Latest Commit Date: **2017/06/20** - Work is still ongoing to deliver a new version.
- Number of main contributors: **1** - It seems to be mainly a one-man project, which is a problem. Some other people have contributed little bits of code, but not to the extent the main contributor has.
- Open issues ratio: **0.0534653465** - With 505 issues opened by many various actors (and not just the main contributor as milestones) since the project's arrival on Github, only 27 remain today, which is a very, very good. Excellent reactivity.

Good project health overall, though a shame that it seems to have only one main developer behind it.

Company backing

- Company Support: **excellent** - the project is backed by a well-established company (seven years of existence), and they offer a separate enterprise version with more features. Support is available to companies through the possibility to buy block hours for assistance, and also to anyone using the community edition through a rather active Google Group instance.
- Company Adoption: **good** - the corporate page of the project gives 2 comments by supposed customers, no success story using sitewhere was found but there were a few press articles talking about it like this one. It can however be infered that if the company has existed for 7 years, it means that they have found clients that appreciate their services.

Good company backing overall, though it's a shame they don't communicate a bit more about their customers.

Documentation

- Currentness: **good** - From the short time spent in the documentation, it seems it has been updated to match the latest release (as its number is written on the documentation homepage), however, in the absence of dating on all pages, it makes it more difficult to say if everything has been updated or not : for that, you have to go to the documentation Github Repository, which turns out to show that the documentation is rather up to date, since the oldest modifications are three month old.

- Adaptability: **excellent** - The documentation is well separated between user and developper use cases. Explanations are clear, with lots of screenshots and appropriate commands where needed. The README.md on the Github Repository is crystal clear. Different levels of documentation are provided, like architecture, technologies, usage, code structure, ...

Good documentation overall, does make you want to use the product.

UI-UX

- Server Management UI: **good** - Powerful and well-documented web administration interface, though its design is a little bit old school.
- Sample applications: **good** - There are a few example applications for ios, android and webapps but not so many examples for hardware.

The proposed UIs seem good overall, but it would take a deeper examination to fully conclude on their capacity.

Security

- Statements: **average** - There is no open commitment to a secure system, on none of their websites. No explanation of the security measures they have taken, except that they use the Spring Frameworks integrated capabilities (which is good). No mention about the cryptographic means they use either.
- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

There is no apparent concern about security on their websites, which is certainly not good. Furthermore, there is apparently an SSL configuration problem with their corporate website, which is not a good sign.

Connectivity/Flexibility

- Number of compatible hardware platforms: **4** - Android, Arduino, Raspberry Pi, and Generic Java-capable board (Cf. dedicated documentation page). Portentially highly compatible with many other platforms, but since no detail is given, we may assume with some reserves that not so many have actually been tested.
- Number of supported protocols: **8** - MQTT, AMQP, OpenWire, XMPP, HTTP REST requests, WebSocket, Hazelcast, Stomp (Cf. dedicated documentation page).
- Number of SDK implementations: **2** - Android and IOS.
- Modularity: **good** - Design built out of preexisting opensource components. DB apparently easily changeable between MongoDB and Apache HBase (Cf. dedicated documentation page)

Overall, many connectors available, though seem to be a little weak on the hardware side, and lack of a SDK for Desktop applications.

Scalability

- Third-Party scalability: **good** - Sitewhere offers a choice between MongoDB, Apache HBase and InfluxDB, which are three highly scalable Data Storage Technologies.
- Platform scalability: **poor** - It seems a Sitewhere instance has no way to communicate with another and allow load-balancing.

It seems that Sitewhere's scalability claim is only based off the capabilities of its 3rd-parties components.

Arduino/NodeMCU compatibility

- Library: **good** - Quite a lot of work seems to have been put into arduino compatibility, and the library seems very complete (with several example sketches included). However, no particular documentation about the NodeMCU/ESP8266.
- Boilerplate code: **good** - The event/publish/subscribe structure, while being very efficient, requires here quite a lot of boilerplate code just to send a little bit of data.

Quite good Arduino compatibility, but it would really be welcome to have some documentation on its usage with NodeMCU/ESP8266 specifically.

Java

- Server percentage: **81.8%** - The server is mainly just plain Java. Exactly what we need.

2.2 Kaa

Project Identity

Corporate URL	https://www.kaaproject.org/
Documentation URL	https://kaaproject.github.io/kaa/docs/v0.10.0/Welcome/
Repository URL	https://github.com/kaaproject/kaa/
License	Apache-2.0
Main Selling Point	Pluggable architecture

Project Health

- First Release Date: **2014/06/30** - The Kaa Project seems to have been open-sourced from its very beginning.
- Latest Release Date: **2016/10/28** - The latest release dates back to the previous year, and could have us think that the project's activity has significantly dropped. The project has an irregular release cycle, "When it's ready"-Debian-like.
- Latest Commit Date: **2017/06/31** - Actually, it seems that the project is still very much active, and a recently posted video on Youtube shows that they plan to release the 1.0 version during this summer. They seem to have reoriented all of their efforts into documenting/testing the project for the upcoming release.
- Number of main contributors: **10** - There is a reasonable number of main contributors, whose work is rather well-distributed along time. All in all, the project has received contributions from 57 different contributors, which shows a good interest from the IoT community.
- Open issues ratio: **0.0173032153** - A very low ratio, almost equal to zero : over the course of the project, 15893 issues have been opened, and now only 275 remain. It certainly goes to show that they have a proactive behaviour toward bugs. It is to be noted that they don't use Github Issues, but their own Jira instance.

Kaa has a very good project health overall, especially since it has reached a point where the team behind it can trustfully say that they are ready for the 1.0 release.

Company backing

- Company Support: **good** - The company behind Kaa, KaaIoT, seems to be more committed to corporate rather than community support (see this StackOverflow discussion), especially with the upcoming release.
- Company Adoption: **average** - Given that the company has existed for 3 years, has, acccording to its LinkedIn page, between 51 and 200 employees, and that they are still recruiting a lot, we can assume they have definitely found clients to buy one of the many formulas they offer on their corporate website. However, no trace was found on who these clients are, likely because of Non-Disclosure Agreements.

Kaa is supported by a rather big team, and after 3 years of continued existence, we may assume that its company backing is good.

Documentation

- Currentness: **good** - It seems like they have well divided and updated the documentation for each version. Though, it has the same little problem as Sitewhere : you have to go to the repository's documentation folder to check the last edition dates since they are not displayed on the website's pages.
- Adaptability: **excellent** - The documentation itself is very-well separated by role of the reader. You have everything: administrator, hardware programmer, contributor guides. Each of these offer detailed explanations, with plenty of screenshots and code snippets to help you get started.

The overall quality of the documentation is very good, and makes you feel you can easily start using the solution right away.

UI-UX

- Server Management UI: **good** - From what can be seen in the documentation, the administration UI is simple but complete, with a not so modern look, but we can bet that will be heavily upgraded in version 1.0.
- Sample applications: **excellent** - Kaa offers a plethora of sample apps that will help you get started, be it with hardware integration or client (apps) integration.

The user experience is especially good thanks to the many sample applications, and the clearly displayed will the Kaa project has to be extremely compatible.

Security

- Statements: **excellent** - A clear concern for security has been expressed by the CTO of the company in a blog post and is also mentioned in their FAQ and documentation. According to these sources, data communication between the components is secured using with 2048bits-RSA encryption and 256bits-AES signing. Apparently, Kaa also ensures secure data storage by enforcing database-level encryption, as well as tenant-level encryption, which is very good practice. In the FAQ, it is also said that Kaa is fault-tolerant, since the data is automatically replicated across the nodes.
- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

A clear concern for security is expressed, and some of the measures taken in favor of it are clearly explained. However, no independent security audit of the solution has ever been executed.

Connectivity/Flexibility

Note: With the upcoming 1.0 version, connectivity and flexibility is apparently about to skyrocket, since according to the Early Access application page, the platform will turn to a SDK-less, technology independent paradigm, with Out-of-the-box MQTT support.

- Number of compatible hardware platforms: **8** - Kaa provides 8 endpoint SDK for many popular hardware platforms : RaspberryPi, ESP8266, ...
- Number of supported protocols: **1** - Devices must use Kaa's own protocol (described here), based on MQTT/CoAP. For easy implementation, Kaa provides many endpoint SDK for the different platforms.
- Number of SDK implementations: **5** - Kaa provides 5 endpoint SDK for all main OSes : Linux, Android, MacOS, Windows, and finally, a generic SDK written in Java.
- Modularity: **excellent** - it is already possible to change some components of Kaa pretty easily, for example, MongoDB and Cassandra are both supported as NoSQL DBs, but it is claimed that for the 1.0 version "Every component of the platform can be customized or substituted with 3rd party software".

Kaa overall boasts a very good connectivity and flexibility, and it seems they will continue in this direction.

Scalability

- Third-Party scalability: **excellent** - Kaa offers a choice between two NoSQL DBS, MongoDB and Cassandra, and two SQL DBS, MariaDB and PostgreSQL, which are highly scalable, and all its other components, like Spark, are made for scalability.
- Platform scalability: **excellent** - Kaa has been thought as a cluster of server nodes, and uses Apache ZooKeeper to coordinate services. Kaa also includes load-balancing capabilities. (Cf. its Architecture Overview)

Kaa seems to have been truly been built from the ground-up to be fully distributed.

Arduino/NodeMCU compatibility

- Library: **good** - As said before, there is a dedicated SDK for the ESP8266, but not for the NodeMCU, neither Arduino.
- Boilerplate code: **good** - There is quite a little boilerplate C code, but it is well-explained, and it shouldn't be really too hard to work with it.

Relatively good compatibility with the ESP8266, however, the support for Arduino or NodeMCU is lacking.

Java

- Server percentage: **69.9%** - Mainly developped in Java (Netty+Spring), almost all of its third party services are written in Java too.

2.3 Device Hive

Project Identity

Corporate URL	https://devicehive.com/
Documentation URL	https://docs.devicehive.com/docs
Repository URL	https://github.com/devicehive/devicehive-java-server
License	Apache-2.0
Main Selling Point	Many integrations

Project Health

- First Release Date: **2013/09/19** - The first release fo the server software has been made in 2013, though, on the corporate website, the copyright indicates an existence starting in 2012. The project has undergone big changes at its beginning, switching from .NET to Java, according to this article.
- Latest Release Date: **2017/08/04** - The last release is very recent, meaning it still actively supported. The project has an irregular release cycle, "When it's ready"-Debian-like.
- Latest Commit Date: **2017/08/17** - As of the writing of this report, commits are made daily on the development branches: we can thus deduce that the project is still undergoing developments.
- Number of main contributors: **5** - It seems that two of the original contributors of the project do not contribute anymore to the server: maybe they ended up managing it. Anyway, the code base seems rather shared among a few people, which is good. A total of 30 persons have contributed to the project, which means there is a mild interest from the FOSS IoT community.
- Open issues ratio: **0.3434343434** - The project doesn't seem to use Github issues a lot: since its creation on Github, only 99 issues have been created and 34 are still open to this day.

The project seems in overall good health, and still growing.

Company backing

- Company Support: **good** - DeviceHive is supported by DataArt, a company that has, according to LinkedIn, 1001 to 5000 employees. Community support seems rather weak, with only a helpdesk that doesn't seem to have been of much use (only one message), and the sporadic use of Github issues doesn't clue at a very active community exchange.
- Company Adoption: **good** - DataArt has a huge list of clients, and received quite a few awards, but most of this is not directly linked to DeviceHive, which seems to be only one of their many projects. On their success stories pages, there is actually one company that mentions DeviceHive: Wot.io.

Overall, DeviceHive has good company backing, though the community aspect of the project seems lacking.

Documentation

- Currentness: **good** - There is a separate documentation for the older 2.0 and latest 3.3.0 versions, which tends to prove that the documentation is up to date, however, there are no easy means to check the last edition dates since it is not even on Github but on a proprietary service called ReadMe.
- Adaptability: **average** - There is no separation of roles in the documentation, but rather a collection of categories, which goes directly from the "Getting Started and tutorials" to "Specific technical information". There is no page adressing the global architecture.

UI-UX

- Server Management UI: **good** - The server management has a beautiful design, and even comes with an integrated tutorial that easily lets you get started. However, it seems it has rather few capabilities (you can only manage three basic tables: networks, devices and JWT tokens, but there is nothing to configure the server graphically).
- Sample applications: **average** - The only client interface provided is a web interface, based on FreeBoard. Nothing for native development. As for sample apps, there is only one for RaspberryPi and one for ESP8266.

Security

- Statements: **poor** - No statements on security whatsoever. It is not even clearly documented what they use for authentication, encryption and signing. Nothing on the corporate website, nothing in the documentation, except an API for authentification which means there is something, but that's it.
- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

Security is not clearly a concern for the DeviceHive project, which is quite bothering for an IoT project.

Connectivity/Flexibility

- Number of compatible hardware platforms: **2** - Only two platforms have been documented : RaspberryPi and ESP8266.
- Number of supported protocols: **3** - REST API, WebSockets or MQTT. This is the minimum to potentially allow for any kind of device to connect, from OS-enabled ones, to barebone ones.
- Number of SDK implementations: **2** - Python, Node/Javascript. It seems there were many more before (Go, Java,...), but they have now been deprecated.
- Modularity: **average** - Though DeviceHive is built on top of popular third-party FOSS components (ElasticSearch, Apache Spark, Cassandra and Kafka), no choice is left as to what to use.

The connectivity and flexibility of DeviceHive seems in fact rather poor overall.

Scalability

- Third-Party scalability: **excellent** - The scalability of the components named above (ElasticSearch, Apache Spark, Cassandra and Kafka) is very good, as they were all designed for this.
- Platform scalability: **good** - DeviceHive uses a container based service oriented architecture approach, managed and orchestrated by Kubernetes. This takes advantage of the inherent capabilities of Docker, but it is not mentioned how the different DeviceHive instances might interact (as their is no Architecture overview in the documentation!).

The approach DeviceHive takes to scalability definitely is interesting, but also cruely lacks documentation.

Arduino/NodeMCU compatibility

- Library: **average** - No Arduino nor NodeMCU libraries, but there is one for the ESP8266.
- Boilerplate code: **good** - Minimal boilerplate code, since it is javascript.

There are some examples with the ESP8266, however they do not seem to cover much.

Java

- Server percentage: **97.2%** - Almost everything is pure Java, which makes it potentially easily maintainable.

2.4 Zetta JS

Project Identity

Corporate URL	http://www.zettajs.org/
Documentation URL	https://github.com/zettajs/zetta/wiki
Repository URL	https://github.com/zettajs/zetta
License	MIT
Main Selling Point	API-First IoT Platform

Project Health

- First Release Date: **2014/07/23** - The project has started on Github three years ago, which is more or less the same as most of the projects presented here.
- Latest Release Date: **2017/07/07** - The latest release is little more than one month old : we can deduce of this that the project is still maintained.
- Latest Commit Date: **2017/07/07** - No advancements have been made since the last release: maybe is it due to summer holidays.
- Number of main contributors: **3** - Only three people are behind this project, and it seems that one of them as left the project at the end of 2016. The Github organization only belongs to one person, which is not necessarily a good thing.
- Open issues ratio: **0.2565789474** - The project doesn't seem to use Github issues a lot: since its creation on Github, only 152 issues have been created and 39 are still open to this day.

Project health looks greenish overall. It might lack maintainers, which would explain the high open issues ratio.

Company backing

- Company Support: **good** - ZettaJS is backed up by APIGEE, a company that creates API-driven applications, and that is been bought by Google in 2016. Actually they sell Apigee Link, a service that leverages Zetta, adding enterprise support and cloud services. As for community support, there is a relatively active dedicated Google Group.
- Company Adoption: **average** - APIGEE seems to have a lot of big customers, and quite a few success stories. However, none of them seem directly related to Zetta.

Overall good company backing, though we have no clear and direct evidence of companies using Zetta.

Documentation

- Currentness: **average** - Most of the wiki hasn't been updated since last year, thus we can infer that there may be a few elements that may differ with the current release.
- Adaptability: **average** - There is no separation of roles in the documentation, but rather a collection of categories, which goes directly from the "Getting Started and tutorials" to "Specific technical information". It definitely feels more like something oriented for tinkerers, rather than to the many different actors using an IoT platform.

Overall, the documentation doesn't look that good. Particularly, it uses the integrated Wiki function of Github, making it clearly dev-oriented, and ignoring all other actors.

UI-UX

- Server Management UI: **average** - There is a UI but it seems only capable to list devices and access their APIs. No user management, no tenant management,...
- Sample applications: **average** - There are a few example projects that are relatively well-documented.

In terms of UI-UX, ZettaJS seems lacking overall, since its focus is primarily in providing APIs.

Security

- Statements: **poor** - No statements about security except one short paragraph in the documentation, that doesn't explain the security strategy at all.
- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

ZettaJS doesn't express much concerns about security, though it is a key element of IoT.

Connectivity/Flexibility

- Number of compatible hardware platforms: **11** - This is hard number to estimate since Zetta rather use a generic concept of "Driver" for any kind of connection. From the official driver list, the following harware platforms have been spotted: Wemo, Sphero, Spark, BeagleBone, Pinoccio, Pebble, OpenXC, RaspberryPi, XiaoMi, Intel Edison, Google Glass.
- Number of supported protocols: **1** - ZettaJS seems to only understand HTTP requests, though it claims it should be able to bridge other protocols to it, but we couldn't find any such example.
- Number of SDK implementations: **1** - There is only a native SDK for IOS.
- Modularity: **good** - As everything is API, any component can conceptually be added, changed, or removed easily.

The API-driven ideology seems to make connexion to the system very easy, but in fact, to this day, there seem to be no easy way to connect devices that don't run a linux kernel on them.

Scalability

- Third-Party scalability: **poor** - LevelDB doesn't seem to have any sort of distribution capabilities.
- Platform scalability: **poor** - As all the servers can easily communicate and exchange data through HTTP APIs, the project seems rather scalable, but there is no distributed data storage, or integrated load-balancing.

ZettaJS doesn't seem to integrate anything related to scalability.

Arduino/NodeMCU compatibility

- Library: **poor** - No official library provided either for Arduino, NodeMCU or ESP8266, but there is a beginning of an implementation here.
- Boilerplate code: **poor** - No official boilerplate code either, but there is a beginning of an implementation here.

No clear support of Arduino/NodeMCU in ZettaJS.

Java

- Server percentage: **0%** - ZettaJS, is, as the name indicates, a full JS solution. No Java here.

2.5 Parse

Project Identity

Corporate URL	http://parseplatform.org/
Documentation URL	http://docs.parseplatform.org/
Repository URL	https://github.com/parse-community/parse-server
License	BSD
Main Selling Point	NodeJS Based

Project Health

- First Release Date: **2016/01/28** - The open-source version of the Parse platform has only been released one year ago, when the company behind it decided to close.
- Latest Release Date: **2017/07/03** - Even though the service closed in January 2017, the project still gets regular releases, the latest one having been made this summer.
- Latest Commit Date: **2017/08/12** - The project is still undergoing changes, though the graphs are pretty clear on the fact that development is winding down.
- Number of main contributors: **2** - Among the 159 contributors, the project is actually only developed by two persons. Although, there are 34 pull requests waiting for integration, and the many forks (over 3000 thousands) indicate that quite a few people are interested in keeping the project afloat.
- Open issues ratio: **0.0742424242** - With 2640 issues created since the project's creatino, and only 196 remaining open today, it is an overall very good ratio, that shows that there is a real interaction with the community, and great reactivity.

It's hard to give an overall appreciation of the project's health. Indeed, one could be tempted to dismiss it as something that is just about to become abandonware, but the seemingly big community around it might keep it going properly.

Company backing

- Company Support: **poor** - Since the company behind the project seems to have kind of dropped it, there is no longer company support. The community seems however relatively active on StackOverflow and Github Issues. An unofficial user community website has also been created.
- Company Adoption: **good** - It seems quite a number of companies were using Parse as a service before, and it is very likely that they simply migrated to the open source version for many of them. Especially most of the community projects around parse are maintained by other companies.

It seems that Parse is now mainly a community-governed project, meaning no easy way to get paid support. However, community support seems reactive.

Documentation

- Currentness: **good** - It seems that the documentation has been well kept to date, with most pages having been edited in april/may 2017.
- Adaptability: **good** - There is a separate wiki for developers that might want to contribute and users.

UI-UX

- Server Management UI: **good** - The project includes a dashboard, which looks pretty neat, with a Data browser and an API console. However, it seems that some functionnalities from the ancient Dashboard of Parse.com have not been integrated in the new one, and won't ever be, like Analytics, since they are no longer supported by Parse itself.
- Sample applications: **good** - There is one repository that has been kept up to date with an example application and some IoT-specific examples can be found in the Parse SDK for Arduino repository, and some other examples can be found on the web.

The overall UI-UX is pretty fine, with many resources that are rather easy to find.

Security

- Statements: **good** - It seems that Parse is taking the subject of security rather seriously, are there are still discussions on how to improve it. Parse uses Application Keys for devicec authentication, but other client-side keys can also be added. They also give very detailed instructions on how to properly implement security in applications using the tools they provide. Finally, they recently forwarded a warning on the official blog about a flaw in NodeJS.
- Audits: **non-existent** - There are no traces of a proper audit of the Parse Platform, but some users did voice concerns about the Parse's security model, which have been adressed at the time.

Parse clearly voice their concern for security, however, there is no trace of an audit to confirm the good application of these principles.

Connectivity/Flexibility

- Number of compatible hardware platforms: **3** - Arduino Yún, Unix-based, and RTOS platforms are supported by two projects: Parse Embedded SDK and Parse SDK Arduino.
- Number of supported protocols: **1** - Everything is done through a REST HTTP API.
- Number of SDK implementations: **7** - iOS, Android, Javascript, .NET + Xamarin, MacOS, Unity, PHP, there are many ways to easily inter-connect with Parse. There are also some experimental or community-initiated SDK implementations, like Parse-Swift or a Ruby client.
- Modularity: **poor** - The project is tightly tied to NodeJS and MongoDB, and it take more than just a little work to get it working with other components.

The overall connectivity-flexiblity is good, but more client-focused than hardware-focused.

Scalability

- Third-Party scalability: **average** - NodeJS and MongoDB are two technologies that are known for their capacity to scale.
- Platform scalability: **poor** - It doesn't seem like any sort of inter-instance communication has been devised, nor is there any integrated load-balancing.

Parse looks like it could scale rather well if only it would integrate clear instructions on how to use several Parse servers.

Arduino/NodeMCU compatibility

- Library: **average** - Efficient implementation, and easily imported. However it is only for the Arduino Yun, and not for the NodeMCU or ESP8266 and would require some adpatations.
- Boilerplate code: **good** - Rather straight-forward boilerplate code, with detailed instructions both in the code and in the wiki.

There is an Arduino compatibility, however, it sadly only covers linux-enabled devices like the Yun.

Java

- Server percentage: **0%** - This is a full JS solution, no Java here.

2.6 Blynk

Project Identity

Corporate URL	http://www.blynk.io/
Documentation URL	http://docs.blynk.cc/
Repository URL	https://github.com/blynkkk/blynk-server
License	GPL-3.0
Main Selling Point	Easy app creation

Project Health

- First Release Date: **2015/06/09** - The project started on January 16, 2015 with a successful Kickstarter campaign, and the project was quickly released under a free-software license in september.
- Latest Release Date: **2017/08/07** - The latest release dates back to the month of this writing: we can deduce that the project is still very much active.
- Latest Commit Date: **2017/08/15** - The latest commit is even more recent, thus we can expect other releases in the future.
- Number of main contributors: **2** - Sadly, the project seems to be almost only maintained by two developpers, one on the server-side, the other one on the hardware-side. There are few other contributors other than them (merely a dozen for each of the repositories).
- Open issues ratio: **0.0295420975** - Since the start of the project, 677 issues have been created, and only 20 remain today. This makes for a really low ratio, testifying a very good reactivity to community input.

The project seems very healthy overall, if we overlook the fact that only two developpers are actively contributing to it.

Company backing

- Company Support: **excellent** - The support plans they provide seem very satisfying, and the company seems healthily running, since its business model of providing only the server and hardware libraries as free software, but leaving the application builder closed-sourced seem to have been a rewarding strategy. The main developers are very active on the forum, complementing the support the community already provides really well.

- Company Adoption: **excellent** - It seems that quite a few companies have adopted Blynk, as their homepage shows: Blynk is part of the "Intel IoT Solutions Alliance", is associated with Sparkfun for Educational Hardware,...

Even though the team at Blynk doesn't seem to be very large, they actually look like they offer a very convincing company backing for their product.

Documentation

- Currentness: **average** - It seems the documentation is not very actively maintained, since most of its pages have
  hrefhttps://github.com/blynkkk/blynkkk.github.ionot been updated for months at least. Often, it must be completed by reading the forum like this forum post testifies. Though the community is very active, it doesn't contribute to the documentation directly, and the afore-mentioned answers do not end up being added to it.

- Adaptability: **good** - Blynk is clearly geared toward the hacker/DIY market. Thus, their documentation reflects this, being more developer-oriented. However, the presentation sections are very understandable for anyone, and are well-separated from tutorials and technical matters. A for server administration, all the information can be found on the server's repository README.MD.

Rather fine documentation, however, the lack of direct contributions from the community is a shame, since it is very active, and many posts on the forum would deserve to be properly transformed into documentation.

UI-UX

- Server Management UI: **average** - The Administration UI offers somewhat the same functionnalities as the other solutions, though not as complete as what you would have with Parse or Kaa. The design is simple and minimalist.

- Sample applications: **good** - The provided Blynk app is excellent in terms of functionnalities, and this part would have deserved an "excellent" mention if and only if there also were less functionnal open-source client apps provided.

Good UI-UX experience overall, especially considering the great phone app.

Security

- Statements: **average** - While the corporate website doesn't mention security much, the documentation actually provide a few details as of the security measures taken by the platform, including authentication and end-to-end encryption.

- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

While the project seems to take security in consideration, this is apparently not their main concern, as they sometimes suggest reducing it, which is certainly not a good idea, especially if the connected devices are actuators.

Connectivity/Flexibility

- Number of compatible hardware platforms: **4OO+** - According to this list, The project is compatible with about just any hardware platform !
- Number of supported protocols: **2** - Blynk only speaks the Blynk protocol and HTTP, which are completely defined here for the custom protocol and here for HTTP. Potentially, this can be extended through a bridge to Nodered, which brings in support for other protocols like MQTT, or XMPP.
- Number of SDK implementations: **2** - iOS and Android apps are the only two SDK available at the moment. No official web implementation, nor desktop at the moment. It is to be reminded that these client apps **are not open-source**, and trying to use Blynk without them might be kind of a pain.
- Modularity: **poor** - To this date, there is no componentization of Blynk-Server: it is just one big standalone, and most of the data isn't actually stored in it, it actually acts more like a bridge between the sensors and the app. If we want to store the raw data, two possibilities are offered: PostgreSQL and .csv files, but no way to leverage this stored data seems to be provided at the moment.

Overall, the connectivity/flexibility advantage of Blynk is solely assured by its hardware compatibility. When it comes to SDK or modularity, this is definitely not the case.

Scalability

- Third-Party scalability: **average** - PostgreSQL is definitely capable of scaling, however, since the app does not take advantage of it, its interest stays limited.
- Platform scalability: **poor** - There is no apparent way to distribute data storage, or any integrated load-balancing between server instances.

The scalability of Blynk seems rather limited, at least for the local version. Since on the official website, they claim that their Blynk Cloud Server handles 21 Billion+ requests per month, they must have devised of a way to scale their server application, but it is unclear how.

Arduino/NodeMCU compatibility

- Library: **excellent** - The provided library is highly complete, with the highest degree of arduino-compatibility around here. There is also everything needed for the NodeMCU and ESP8266.
- Boilerplate code: **excellent** - Minimalist and powerful, many examples are given for just about any platform: the library takes care of almost everything.

Blynk boasts the best hardware compatibility out there, and such an ease of integration is simply found in none of the other projects.

Java

- Server percentage: **96.9%** - This is a fully java-driven server, the other traces of code are HTML and CSS, which do not quite count.

2.7 Kapua

Project Identity

Corporate URL	https://www.eclipse.org/kapua/
Documentation URL	https://www.eclipse.org/kapua/documentation.php
Repository URL	https://github.com/eclipse/kapua
License	EPL-1.0
Main Selling Point	Modular

Project Health

- First Release Date: **2017/04/28** - The Kapua project is extremely recent, in comparison to the others that all more or less started in 2013/2014. It started in 2016 as part of the Eclipse IoT ecosystem, according to the date of the first commit.
- Latest Release Date: **2017/08/11** - This date doesn't mean much since it is only the second release of the project, and that it is still in its early beginnings.
- Latest Commit Date: **2017/08/21** - The project is currently under very heavy development. New commits are added every day.
- Number of main contributors: **8** - The project is maintained by approximately 8 persons, which makes for a decent number of people.
- Open issues ratio: **0.335243553** - The high ratio is easily explained by the novel character of the project. What is interesting however is that it is not just people that are working on the project who post issues, but there seem to be a few users already, too.

The project has just been born, and from what we can see, it is clearly going at a steady pace and growing rapidly.

Company backing

- Company Support: **good** - Kapua is supported by three big names : RedHat, Eurotech and Bosch. Redhat, for one, is a very well-known figure of the FOSS world. Of course, the project is also sponsored by the Eclipse Foundation, giving it even more credibility. Support is apparently mainly provided through the project's Github issues, and the staff seems quite reactive to such queries. The community is also active on a forum and a mailing-list. Though, it doesn't look like they have an active dedicated support line like the many projects that were named before.
- Company Adoption: **good** - It seems that the main goal of the companies behind Kapua (along with its sister projects Hono and Kura, is for it to be used internally for all of their IoT projects. Apart from these companies, at the moment, it doesn't seem there are external customers using Kura.

Kapua is backed-up by big names of the digital industry, which is reassuring for the future of the project.

Documentation

- Currentness: **average** - Since the project is very recent, we can assume that the documentation cannot have gotten far behind. Actually, according to the repository, the latest documentation modifications date back one month. Features like the Server Management UI are not in the documentation, so we can guess that this documentation may be lacking. On another note, what is especially pleasing about this documentation is its very powerful and good-looking display allowed by GitBook.

- Adaptability: **average** - The user guide and developer guide are well-separated. There is even a separate Getting Started page. However, in the end, almost only the developer guide is truly interesting, since the user guide is very poor.

Overall, we can assume that the not-so-good documentation is the result of young age of the project.

UI-UX

- Server Management UI: **average** - From the very little we can see on the Getting Started page, it seems there is such a tool, though it seems it is just an old interface originally developed by Eurotech (see copyright 2001-2006 at the bottom of the screenshot), and it is not mentioned in the documentation.
- Sample applications: **poor** - There seem to be a provided Kura simulator, but by no means can this be considered a sample application, rather a development tool. There is also a mention of a "heater app" on the Getting Started page, but it is unclear as to where to find it, and what it does.

Expectedly, since the project is rather recent, its UI-UX is rather poor overall. However, since the project is still under heavy development, we ca only expect it to get better.

Security

- Statements: **average** - The only mentions of the security procedures the project has deployed are on the JWT Security documentation page. This merely hints at the use of encryption, but no details as to how it is enforced is given. A good point is that, thanks to the fact it is an Eclipse project, a proper contact form for reporting security issues is provided.
- Audits: **non-existent** - No mention on the product website, neither found any through a quick search on the web.

Connectivity/Flexibility

- Number of compatible hardware platforms: **1** - Linux-based and Kura-Enabled hardware seems to be the only kind of compatible hardware. This includes the well-known RaspberryPi or BeagleBones.
- Number of supported protocols: **1** - It seems that currently, Kapua is only supposed to connect to MQTT-Kura enabled devices. Actually, the framework seems highly coupled with this other project.
- Number of SDK implementations: **0** - No SDK of any sort has been found.
- Modularity: **poor** - It seems that aside from being able to switch between two database systems, MariaDB or H2 SQL Server, the project is rather monolithic, and does seem easy to fiddle with.

In the end, it seems that Kapua is far from complete, and its overall adaptability is rather poor.

Scalability

- Third-Party scalability: **average** - Apart from the databases, the docker images are appreciated, but that's it.
- Platform scalability: **poor** - There doesn't seem to be any effort geared toward scalability (the key-word doesn't even appear on the project's corporate page), no load-balancing, no data/computation distribution,...

Kapua doesn't seem to particularly shine in terms of scalability.

Arduino/NodeMCU compatibility

- Library: **non-existent** - There is no support for Arduino/NodeMCU/ESP8266 whatsoever.
- Boilerplate code: **non-existent** - Thus no boilerplate code.

Kapua has no support for Arduino/NodeMCU/ESP8266 at all.

Java

- Server percentage: **91.8%** - Thanks to its status as an Eclipse project, Kapua is almost fully written in Java.

2.8 NodeRed

Project Identity

Corporate URL	https://nodered.org/
Documentation URL	https://nodered.org/docs/
Repository URL	https://github.com/node-red/node-red
License	Apache-2.0
Main Selling Point	Flow-based programming

Project Health

- First Release Date: **2013/10/16** - The project, as most of the others that are compared here, was created around 2013/2014, which makes it one of the long-standing contestants.
- Latest Release Date: **2017/07/23** - Since the first release, 54 other releases were made, the latest one dating back from one month before this review. On top of that, the release cycle is highly regular. We can therefore assume it is still an active project.
- Latest Commit Date: **2017/08/21** - The work did certainly not stop at the last release, and another one is in the making, as the latest commits prove it.
- Number of main contributors: **2** - The project's code base is mainly maintained by two persons, though 56 contributors in total have brought their little piece to it.
- Open issues ratio: **0.1063829787** - Over the course of the projects, 940 issues were created, and 100 remain to this day. The ratio is rather good, but in the light of the age of the projects and its extensive use of issues, it seems that the team may require more help treating all the queries.

The project is in rather good health, aside from the fact the two main developers seem like they could use some help.

Company backing

- Company Support: **good** - Node-RED was originally developed as a side-project by two members of the IBM's Emerging Technology Services team and is now a part of the JS Foundation, of which IBM is one of the founding members. It is unclear whether IBM still finances the project, but it is clear that its main two developpers do commit a lot of their time to the project, and when they are not available, the very strong community fills in the GAP for support. However, there seem to be no clear company support offer, like a direct line to the devs through email or phone.

- Company Adoption: **good** - According to this blog post, companies such as SenseTecnic, AT&T and Red Ant provide Node-RED services. Even though the software still hasn't released a 1.0 version, it seems rather trusted and well-known in the IoT industry. It also seems they have a partnership with the RaspberryPi foundation, since Node-RED is now available by default on every new RaspberryPi device.

Even though the project's governance is clearly willed to be community-driven, the company-backing doesn't look half-bad, with some big names trusting the project.

Documentation

- Currentness: **good** - According to the Github repository of the website, the documentation is actively maintained, the latest changes having been made only 5 days before the writing of this review, and the oldest only 6 months old.
- Adaptability: **excellent** - The documentation is **very-well organised**. User and developer resources are clearly separated, all very complete and accessible.

The documentation looks excellent overall. Add to this the many resources that have been devised by third-parties, like the ones of Sense Tecnic, and you have no reason not to easily get started.

UI-UX

- Server Management UI: **excellent** - Actually, one of the main forte of the project is clearly its embedded UI that enables dynamic Flow-Programming. Everything is done through simple boxes, "nodes", linked together to create new functionnality. This workflow feels just right for the IoT, which is actually just that: connecting boxes together.
- Sample applications: **good** - Actually, since Node-Red can be run on Edge devices, it is in itself a sample application, and can be used with a cloud instance. There is also a provided web-based dashboard with many widgets from which to choose and compose your own application. However, an example Android or iOS native app couldn't be found.

In terms of UI-UX, Node-RED clearly tops any of the other solutions.

Security

- Statements: **good** - Most of security measures taken in Node-RED are described on a dedicated documentation page. Though it seems that by default, Node-RED is insecure, all the steps to properly secure it with authentication and encryption are described on this page. However it doesn't seem there is any kind of storage-encryption.
- Audits: **poor** - While it seems to be the only project of the entire list that got a security audit, the one we found is rather poor since automated. Actually, it does have the merit to detect some missing HTTP security features, but many of the detected "problems" are actually false alerts.

It seems that the people behind Node-RED do care about security, though there seems to be some room for improvement.

Connectivity/Flexibility

- Number of compatible hardware platforms: **17+** - According to the list of nodes in this repository, there are already 17 hardware platforms that benefit of a node, but since it doesn't even include the arduino one, we can safely assume there are many more that are reported in this list.

- Number of supported protocols: **2** - It seems that Node-RED is essentially capable of HTTP and MQTT interactions (actually, its roots come from this second protocol).
- Number of SDK implementations: **2** - Let's say that the Node-RED interface makes one since it can be used on phones as well. To that we can add the provided web-based dashboard.
- Modularity: **excellent** - Thanks to its "node"-paradigm, Node-RED can easily integrate just about any component, already including three DBs (LevelDB, MySQL, SQLite) for example.

Simply put, Node-RED is a model of connectivity/flexibility/modularity.

Scalability

- Third-Party scalability: **good** - The nodes that integrate third-party components offer a very good scalability by themselves.
- Platform scalability: **good** - The platform does not include scalability elements (planned for version 0.20) like load-balancing and the such. However, it seems it is already easy to make to NodeRED instances communicate together (see links in the UI-UX part).

Arduino/NodeMCU compatibility

- Library: **good** - It doesn't seem like there is a specific node and official documentation for the NodeMCU or the ESP8266 like there is for the Arduino, but it seems that we can make do with some third-party tutorials using generic nodes like this one, or this one.
- Boilerplate code: **excellent** - Actually, since its all flow-programming, there is no boilerplate code, which is very much appreciated !

Java

- Server percentage: **0%** - NodeRed is a full Javascript/NodeJS project.

3 Summarized analysis

In order to ease the reading of the comparison tables, you will find below the color-correspondance table for appreciation-type criteria, from worst to best :

non-existent	poor	average	good	excellent

For the other types of criteria, comparatively bad results are indicated in red, and good ones are indicated in green. If they are in-between, they are indicated in orange.

If you haven't read the detailed analysis or the argumentation behind the evaluation criteria, please consider doing so if you think one of the values below surprises you.

3.1 General criteria

Project Health

Solution	First Release	Last Release	Last Commit	Main Contr.	Open Issues Ratio
SiteWhere	2014/03/06	2017/06/19	2017/08/11	1	0.0534653465
Kaa	2014/06/30	2016/10/28	2017/06/31	10	0.0173032153
DeviceHive	2013/09/19	2017/08/04	2017/08/17	6	0.3434343434
ZettaJS	2014/07/23	2017/07/07	2017/07/07	3	0.2565789474
Parse	2016/01/28	2017/07/03	2017/08/18	4	0.0742424242
Blynk	2015/06/09	2017/08/07	2017/08/15	2	0.0295420975
Kapua	2017/04/28	2017/08/11	2017/08/21	8	0.335243553
NodeRed	2013/10/16	2017/07/23	2017/07/21	2	0.1063829787

According to the table, we can see that most of the projects started around the same period, in 2013/2014, even Parse (before 2016, it was closed-source software). Only Blynk arrived a little bit later, and Kapua is an entirely new project.

All the projects seem still alive and well, as show their latest release and commit dates, though for Kaa, it is to be noted that they stopped their usual release cycle for the preparation of their 1.0 version, which should come by the end of summer 2017.

Kaa, DeviceHive and Kapua seem to have large team working on their core component, while the other projects are mainly maintained by 2 or 3 people. Worst of the list is SiteWhere, with only one main developer maintaining it: if he were to leave the project, it would stop alltogether.

DeviceHive, ZettaJS and Kapua have a rather high open issues ratio, for different reasons: for Kapua, it is because the project is still in its early phase, but for DeviceHive and ZettaJS, it seems to come from a lack of reactivity from the developper team.

Overall, the three projects that seem to be in best health are Kaa, Blynk and Node-RED.

Company Backing

Solution	Company Support	Company Adoption
SiteWhere	excellent	good
Kaa	good	average
DeviceHive	good	good
ZettaJS	good	average
Parse	poor	good
Blynk	excellent	excellent
Kapua	good	good
NodeRed	good	good

All projects have some kind of company-backing, ranging from dedicated companies to big companies that also develop many other solutions. However, Parse is a project that is clearly loosing its speed since the company behind it closed: it is now fully community-driven, which doesn't seem to be the best for its continuity.

Documentation

Solution	Currentness	Adaptability
SiteWhere	good	excellent
Kaa	good	excellent
DeviceHive	good	average
ZettaJS	average	average
Parse	good	good
Blynk	average	good
Kapua	average	average
NodeRed	good	excellent

All projects have a relatively decent documentation, aside from ZettaJS and Kapua. ZettaJS's documentation hasn't been updated in a long while and is not as well-organised as the one of the others, and Kapua's lack in documentation can easily be explained by the young age of the project.

UI-UX

Solution	Server Management UI	Sample applications
SiteWhere	good	good
Kaa	good	excellent
DeviceHive	good	average
ZettaJS	average	good
Parse	good	good
Blynk	average	good
Kapua	average	poor
NodeRed	excellent	good

In terms of UI-UX, all projects come with rather decent proposals, except Kapua, which again, is probably mainly due to its young age.

3.2 IoT-specific criteria

Security

Solution	Statements	Audits
SiteWhere	average	non-existent
Kaa	excellent	non-existent
DeviceHive	poor	non-existent
ZettaJS	poor	non-existent
Parse	good	non-existent
Blynk	average	non-existent
Kapua	average	non-existent
NodeRed	good	poor

Security is the criteria where the difference is clearly made between the solutions.

DeviceHive and ZettaJS don't give much clues about the security procedures they implement, which may hint at a bad design, and Sitewhere, Blynk and Kapua, while they do give details about their security architecture, these are not quite as satisfying as those found for Parse or Node-RED. Kaa definitely shines most among its brothers in terms of security architecture, and truly shows an appropriate interest for security concerns.

However, none of the solutions have ever been submitted to a true security audit realized by an independent entity. Which is quite stunning (and not in a good way) considering the importance of security in IoT systems, especially if they contain actuators.

Connectivity/Flexibility

Solution	Hardware platforms	Supported protocols	SDK implementations	Modularity
SiteWhere	4	7	2	good
Kaa	8	1	5	excellent
DeviceHive	2	3	2	average
ZettaJS	11	1	1	good
Parse	3	1	7	poor
Blynk	400+	2	2	poor
Kapua	1	1	0	poor
NodeRed	17+	2	2	excellent

The connectivity/flexibility of a solution is one of the hardest things to assess. Indeed, each solution provides its own way of communicating with third-party devices or softwares, with their very own vocabulary and paradigm. Thus the estimations given here need reading the detailed analysis to fully understand their meaning, and why, for example, Blynk and Kaa both deserve the "excellent" appreciation for their number of compatible Hardware Platforms, though one boasts more than 400, and the other, "only" 8.

In terms of connectivity, the platforms that shine most are: Blynk for its concern about hardware, Sitewhere for its support of a great many communication protocols, Kaa and Parse for their excellent number of provided SDK, and Node-RED + Kaa for their excellent Modularity.

Overall, Kaa seems to be the best option in terms of connectivity, flexibility and modularity.

Scalability

Solution	Third-Party scalability	Platform scalability
SiteWhere	good	poor
Kaa	excellent	excellent
DeviceHive	excellent	average
ZettaJS	poor	poor
Parse	average	poor
Blynk	average	poor
Kapua	average	poor
NodeRed	good	good

Though more or less all the platforms presented here claim to be scalable, only a few truly exposed what made them so. While Node-RED and DeviceHive seemed to offer a decent reflexion on the question, **only Kaa presented a full-fledged approach to scalability, with integrated data distribution, inter-instance communication, and load-balancing**.

3.3 Application-specific criteria

Solution	Arduino Library	Arduino Boilerplate Code	Java Server Percentage
SiteWhere	good	good	81.8
Kaa	good	good	69.9
DeviceHive	average	good	97.2
ZettaJS	poor	poor	0
Parse	average	good	0
Blynk	excellent	excellent	96.9
Kapua	non-existent	non-existent	91.8
NodeRed	average	excellent	0

Finally, come the very subjective criteria on Arduino compatibility and Java composition.

Since ZettaJS, Parse and Node-RED are full-javascript solutions, they will not be retained as final solutions here. The lower Java percentages for Sitewhere and Kaa are simply explained by the presence of HTML for the web-ui, and in the case of Kaa, by the presence of Client-side code in the main project repository, where all the server code is situated.

For the others, only SiteWhere, Kaa and most importantly, Blynk offer a decent Arduino/NodeMCU/ESP8266 support. The other ones either provide a lacking implementation, or as for Kapua, no implementation at all.

Conclusion

Finally, this report allows us to conclude on several things:

- There are quite a few very decent FOSS IoT middleware cloud platforms out there, and some of them are actually used for production already, mainly by the companies that develop them.
- Each of these solutions has a different approach and specific functionnalities, that may not have been all described here, but the aim of this review was to find common points of comparison.
- While the final choice will go for a Java-based solution in the OCCIware case, it does not mean at all that the NodeJS-based solutions are uninteresting, especially Node-RED, which brings in a passionating abstraction layer.
- Kapua might become a very interesting choice in the future, but the project is far from mature right now.
- DeviceHive and SiteWhere are two very interesting solutions, but their apparent lack of real scalability and concern for security make them less appealing in the end.

In the end, the final choice for the OCCIware project would clearly be the Kaa Project, if not for the time and resource constraints of the LinkedData Demonstration: if the 1.0 version were available at the time of this writing, and it would be possible for the project to run on slightly lower base specifications (minimum 4Gb of RAM), it would definitely have been the go-to choice. So, **while Kaa is the platform that would be recommended at the end of this report**, we shall go with a Blynk instance for the meantime, notably for its very good prototyping qualities that will allow us to more quickly complete the demonstration.