Research Static Analysis Tools (SAT)

This document contains the results of the SAT research.
While researching, the results of other researches (that were performed within the same sprint) were taken into account.
The following results were the most important ones for this research:
- For the framework research: we're using TypeScript, so it would be nice to have static analysis tools that can be used for it.
- For the IDE research: we're using WebStorm, so the ability of integration in WebStorm is a must.

Suggestions ('major' tools for static analysis):

- *TSLint - 'An extensible linter for the TypeScript language'*
  (integration with Webstorm)
  https://palantir.github.io/tslint/, https://github.com/palantir/tslint
  Description: TSLint is a SAT that enforces a set of rules during software development. These rules are there mainly for assuring a consistent style throughout the development. Therefore TSLint is quite comparable with the Eclipse plug-in 'Checkstyle', which is used throughout software development in Java for assuring a consistent programming style. For the full set of rules, please check the link above.
  + Specially designed for TypeScript
- *JSHint – 'A Javascript code quality tool'*
  (integration with Webstorm)
  http://jshint.com/ , https://github.com/jshint/jshint
  Description: JSHint is a very well-known SAT for Javascript software development. JSHint finds bugs and potential problems via a set of rules (just like TSLint, but keep in mind that TSLint is there for TypeScript and JSHint is there for JavaScript in general). JSHint is seen as the 'more flexible' SAT version of JSLint, containing a new load of rules and removed antagonistic rules (rules that can conflict with each other).
  + ~ Larger set of rules and it's used a lot
  - Maybe 'too' general for the project, when compared with TSLint
- *JSLint – 'The Javascript code quality tool'*
  (integration with Webstorm)
  http://jslint.com/ , https://github.com/douglascrockford/JSLint
  Description: JSLint is also a very well-known SAT for Javascript software development. It's quite similar to JSHint, but more or less the 'predecessor' (containing antagonistic rules).
  - Compared to JSHint, JSLint might not be the 'best' SAT to use
- *ESLint – 'The pluggable linting utility for Javascript and JSX'*
  (integration with Webstorm)
  http://eslint.org/ , https://github.com/eslint/eslint
  Description: ESLint is the even more 'flexible' version of JSHint and provides relatively even more options for its rules. Besides, ESLint is a relatively new SAT.
  + Compared to JSLint, ESLint might be a 'better' SAT to use (together with JSHint)
  + Compared to JSHint, even more options for the rules, for more flexibility

Recommendation: **TSLint**, as our project will be developed in TypeScript, it is useful to use a SAT that focuses its metrics for this language. The alternatives that are proposed above are used for development
in Javascript in general and thus could have rules that might not be useful for the language we're using.

Finally some *'minor tool' suggestions* for static analysis:

- *Dependo*
  https://kenneth.io/blog/2013/04/01/visualize-your-javaScript-dependencies-with-dependo/,
  https://github.com/auchenberg/dependo
  Description: Dependo visualizes internal dependencies by a directed graph.
  - Dependo is still in progress, according to the notes
  + It's simple
  + Produces a graph that's interactive
- *MaDGe (Module Dependency Graph)*
  https://github.com/pahen/madge
  Description: Very similar to dependo; it also visualizes dependencies. This is useful for finding circular dependencies within the code.
  + Compared to Dependo, it a more advanced and documented tool for the visualisation.
  - It requires another tool: GraphViz

Recommendation: **MaDGe**, as the tool is relatively more advanced and we can use the visualisation throughout the project for taking internal dependencies into account.