

**Kauno technologijos universitetas**

INFORMATIKOS FAKULTETAS

**T120B162 Programų sistemų testavimas**

IFF-6/11 Nerijus Dulkė

Data: 2019.12.09

KAUNAS

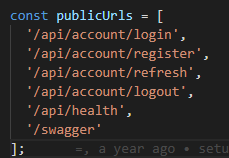
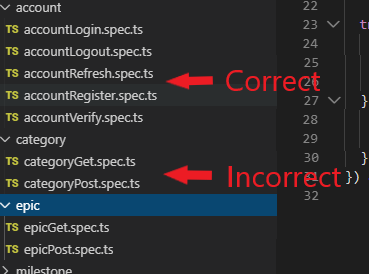
2019

# Task 1. Code review

Code review checklist:

* No unused variables
* Names are consistent
* Consistent code style
* No duplicated logic
* Code is readable and clear upon initial reading
* Code that is unclear has an effective comment
* Unexpected errors are handled
* No commented-out code
* At least 80% coverage on new code
* All tests are passing
* Tech-debt registered to the log

Found issues after code review:

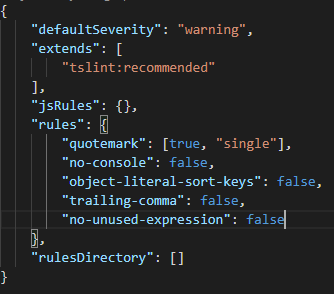
* All controllers implementing CRUD actions have a lot of duplication. Can be split into more generic service or helper.
* Handling promises should be consistent (either *.then()* or *async/await*, not both).
* Authentication middleware should have a more robust way of defining public URLs.  
  
* Test files names are not clear.  
  
* Unresolved *TODO* tag in *app.ts*  
  

# Task 2. Static code analysis

For static code analysis TSLint was used (ESLint version for TypeScript). Since this tool was integrated into development environment during the development process, all issues raised are fixed during the development of features.

All recommended rules with a few exceptions are used for analysis (available from <https://github.com/palantir/tslint/blob/master/src/configs/recommended.ts> ).

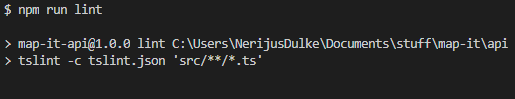
TSLint config file:



Command to show existing errors:



Command output:



# Task 3. Custom rule

A custom TSLint rule was added to prevent throwing generic *Error* types. Only extended error types can be thrown to be more understandable, for example custom *HttpError* can be thrown.

Custom rule code:

import \* as Lint from 'tslint';

import \* as ts from 'typescript';

export class Rule extends Lint.Rules.AbstractRule {

  public static FAILURE\_STRING = 'no generic errors';

  public apply(sourceFile: ts.SourceFile): Lint.RuleFailure[] {

    return this.applyWithWalker(new NoGemericErrorsWalker(sourceFile, this.getOptions()));

  }

}

// tslint:disable-next-line: max-classes-per-file

class NoGemericErrorsWalker extends Lint.RuleWalker {

  public visitThrowStatement(node: ts.ThrowStatement) {

    if (node.expression.getChildCount() > 1 && node.expression.getChildAt(1).getText() === 'Error') {

      this.addFailure(this.createFailure(node.getStart(), node.getWidth(), Rule.FAILURE\_STRING));

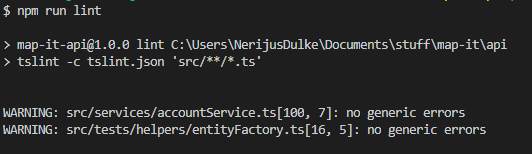
    }

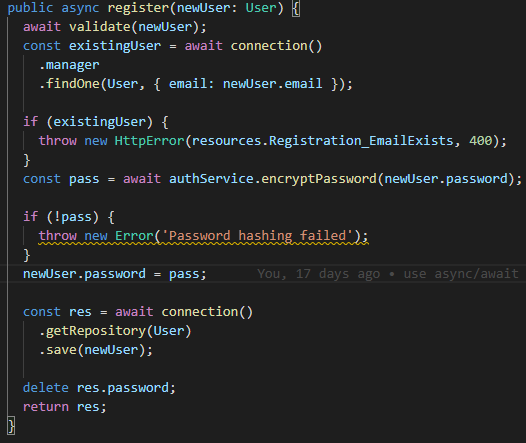
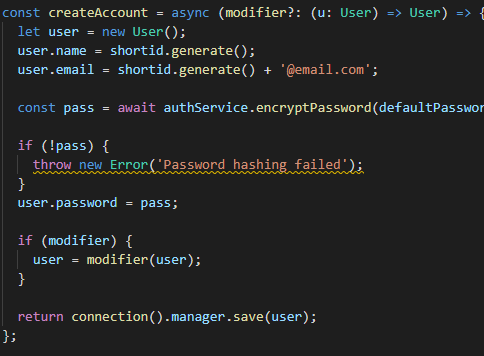
    super.visitThrowStatement(node);

  }

}

Output after running TSLint with custom rule:



Two issues were found:

# Conclusion

Found some tech-debt issues that cause bad code readability and maintainability. Project is constantly analyzed during development with static code analysis tool. Also learned how to define mt own custom rule.