**Configuration Management Plan**

**Project: was-tun.at**

Content

[Versioning 2](#_Toc468130938)

[Methods & Tools 2](#_Toc468130939)

[Organization of Artifacts 3](#_Toc468130940)

[Variant Management 3](#_Toc468130941)

[Version Management 3](#_Toc468130942)

# Versioning

## Methods & Tools

As our versioning method we decided to use git. So what Tools do we use? First of all, one of the most important is our repository which is hosted on bitbucket.org and uses git as mentioned before. In combination to Bitbucket the software SourceTree is used for managing our local repositories and Project Files.

Tools we use for perfect versioning:

|  |
| --- |
| Bitbucket.org |
| Git |
| Source-Tree |

Additional tools are Visual Studio Code and IntelliJ which have git implemented as feature to manage the repository instantly while coding in Front/-Backend.

Without methods, there would be no right versioning of our project. The following paragraphs will be a brief explanation of our Methods.

When the project started we instantly created two repositories, one for Backend and the other one for Frontend Source.

Within the Frontend Repository we created a branch called “project-documents” where all our documents are stored, like this one.

In short, all files which belong to our project are under version control. (Source and Documents) And we strictly hold in the git-flow. More about in Variant Management.

# Organization of Artifacts

In our Frontend repository there are the following “Management” Folders:

1. Project Planning
   1. Was-tun Schedule.excel
      1. Containing the current work of the team-members.
   2. Requirements.docx
   3. Proposal.docx
   4. Configuration ManagementPlan.docx
2. “Source-Code” is in the folder “app”
3. Release-Docs
   1. Changelog.docx
      1. Containing all features to date and the changelog.

# Variant Management

There’s nothing more to say than “Git-Flow”. We strictly work and follow the Git-Flow given by git.

Structure of Branches:

* Master
  + Never push into master except a new final version is reached in develop
* Develop
  + This branch should always be “finished” or able to go “online”.
* Feature/<feature>
  + New feature for the project. After finishing the feature it’ll be merged back to develop.
* Release/<releaseVersion>

# Version Management

First of all there is a structured name like v1.0.1. For each update or bug-fix, we increase our version number in the third place to v1.0.2 for example. For a new feature, the version goes up in second place to v1.1.1. If a final version is finished, the version number changes in first place like v2.0.1

Each final version will also be called after “Kulinarische Österreichische Köstlichkeiten” like: “Grestl”, “Kaiserschmarrn”, “Beischl”, “Schnitzl”, “Kasnockn” and so on..