Devdoc-Swissknife-Ru

Николай Гнитеев

2021-09-30

Contents

Вступление

Этот проект демонстрирует подход к созданию документации на разработку с помощью R Markdown и Kroki.

Данный подход позволяет создавать документацию в виде файлов PDF, презентаций, сайта документтации по шаблону gitbook и в некоторых других форматах, используя для этого текстовые файлы с синтаксисом markdown (Pandoc flavor) и дополнительными вставками кода на языке R, а так же текстовые описания разного рода диаграмм.

Целью этого проекта является создание документации, максимально используя для этого текстовый формат, в т.ч. для описания графических диаграмм, но при этом без излишних переусложнений.

За счёт текстового описания диаграмм, с ними проще работать в системах контроля версий, получать разницу между версиями, добавлять к ним коментарии, делать обзор изменений и т.п.

Часто может оказаться и так, что создание диаграмм, а тем более их развитие быстрее делать в текстовом виде, а так же делать такие вещи как применение и/или рефакторинг цветовых схем. Поскольку зачастую тектовое описание диаграмм это больше опиисание некой модели, чем описание графических элентов,то при редактирвоании таких описаний не требуется тратить усилия на то чтобы в графическом виде взаимосвязи сохранялись, т.к. это будет происходить автоматически, покуда не изменились названия объектов модели.

Уже один только R Markdown сам по себе является мощным и настраиваемым инструментом, который можно подстроить на формирование документов в том виде, в котором это вам требуется. Конечно для этого в начале придётся приложить определённые усилия, но в конечном итоге они окупятся.

Чтобы на выходе были великлепные файлы PDF нужен и

> соответствующий класс для Latex. Пока что у меня такого нет, поэтому в данном проекте PDF файлы выглядят довольнно-таки просто. Но надеюсь, что рано или поздно, появится что показать и с этой стороны.

Для визуализации диаграмм используется локальный сервер Kroki, не требующий подключения к сети Internet, что позволит не беспокоится за сохранение конфиденциальности данных.

Помимо демонстрационных целей этот проект так же может быть использован как шаблон для формирования вашей собственной документации. В этом разделе данный вопрос рассмотрен подробнее.

Все инструменты для создания документации запускаются из контейнеров docker, что упростит встраивание этого подхода в ваш конвейер CI/CD.

Структура оргнизации документации, а так же некоторые файлы взаимствованы из R Markdown book

---

В проекте приводится несколько примеров диаграмм для наиболее частых случаев, требуемых в разработке. По ссылкам на проекты Вы сможете найти что-то и для менее типовых задач.

> Для большинства примеров использован код с сайта Kroki

Полезные ссылки:

Список поддерживаемых визуализаторов диаграмм - https://kroki.io/#support

Немного больше примеров диаграмм - https://kroki.io/examples.html

R Markdown: The Definitive Guide - https://bookdown.org/yihui/rmarkdown/
от авторов R Markdown и созданное с помощю R Markdown

R Markdown: Cookbook - https://bookdown.org/yihui/rmarkdown-cookbook/

Неплохая методичка по R Markdown - https://rmd4sci.njtierney.com/

R Markdown cheatsheet - https://raw.githubusercontent.com/rstudio/cheatsheets/master/rmarkdow

R Markdown reference - https://rmarkdown.rstudio.com/docs/reference/index.html

Keenwrite - редактор с поддержкой предварительного просмотра всех диаграмм!

Chapter 1

Простые примеры визуализации диаграмм

Поскольку основной фокус этого проекта на встраивании диаграмм, описанных в текстовом виде, в файл R Markdown (т.к. остальное в большинстве случаев смогут обеспечить возможности самого R Markdown), для начала приведена пара простых примеров для иллюстрации самого принципа.

1.1 Пример с описанием диаграммы непосредственно в тексте файла документации

Вставка такого кода в файл документации:

```
```{r echo=FALSE, results='asis'}
  to_diagram("graphviz", "Hello World",
  "digraph G {Hello->World}"
  )
```
```

добавит такую диаграмму:

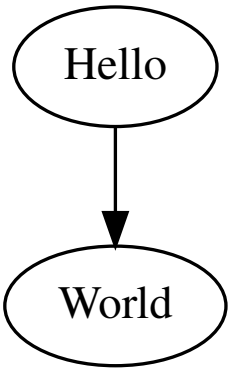

Figure 1.1: Hello World

1.2 Пример с вставкой диаграммы, описанной в отдельном файле

Вставка такого кода в файл документации:

```
```{r echo=FALSE, results='asis'}
  to_diagram("erd", "Entity Relation", src="../diagrams/examples/project.erd")
```
```

добавит такую диаграмму:

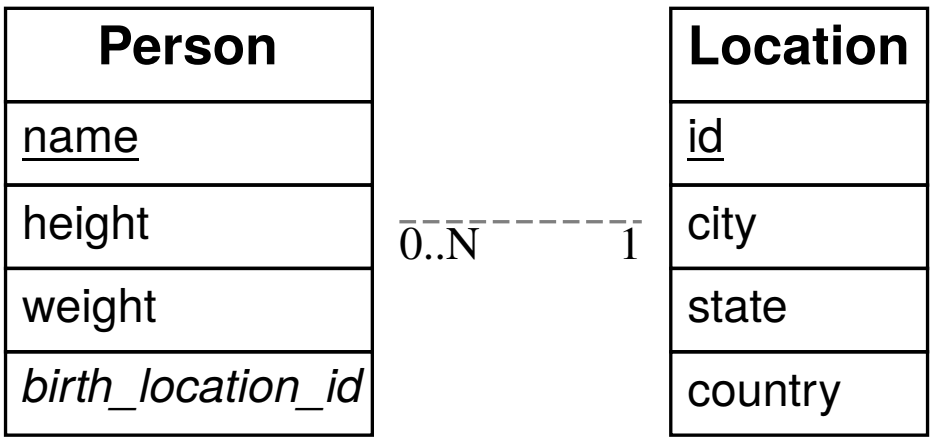

Figure 1.2: Entity Relation

Содержимое файла **diagrams/examples/project.erd** с описанием диаграммы:

```
[Person]
*name
height
weight
+birth_location_id

[Location]
*id
city
state
country

Person *--1 Location
```

1.3 Больше примеров

Если Вам всё ещё интересно, больше примеров приведено в этой секции.

Chapter 2

Использование проекта в качестве шаблона и создание собственной документации

Чтобы воспользоваться данным проектом как шаблоном для создания собственных документов, сделайте следующее:

1. Импорт проекта и подготовка

   У Вас уже должны быть установлены `docker` и `docker-compose`

- Импорт ветки main из этого репозитория или создание форка: https://github.com/Godhart/devdoc-swissknife
  Исходные файлы для создания документации располагаются в папке `docs_src`.
- Создайте новуй папку внутри `docs_src`. Рекомендуемый шаблон для имени папки: `doc-<subject>`
- Скопируйте все файлы из `docs_src/docs-template` в только что созданную папку
- В скопированных файлах замените следующий текст актуальными значениями: `<Author Name>`, `<author>`, `<repo>`, `<Document Title>`, `Document_Title`, `<Document Description>`

Не пропустите замену текста `Document_Title` и в файле `.gitignore`, который так же будет среди них.

- Если вам захочется избавиться от документации devdoc-swissknife,

которая тоже будет в этом репозитории:

  - Удалите папки с именем по следующему шаблону `docs_src/devdoc-swissknife-*`
  - Очистите папку `docs_src/diagrams`.

- Поправьте `docs_src/Makefile` как Вам требуется (для начала - просто воспользуйтесь шаблоном, который описан в этом файле).
- Если вы до этого не создавали образ docker `devdoc-swissknife` - сделайте это сейчас. Просто выполните файл `make_docker.sh` из папки `docker`.
- Можно попробовать создать документацию - выполните файл `make_docs.sh`. Результаты сборки появятся в папке `docs_out/doc-<subject>` если вы следовали шаблону из Makefile.
  Обратите внимание, что папка `docs_out` и всё её содержимое игноирурются в git.

  ВНИМАНИЕ: зачастую в случае ошибок сборки документа папка с исходными файлами документов м.б. засорена временными файлами, их имя будет соответствовать значению поля `book_filename` из файла `_bookdown.yml` и их наличие может сломать последующие сборки.
  Обычно эти файлы удаляются при сборке и не составляют проблем, но иногда может потребоваться удалить их самостоятельно.
  К примиеру таким случаем м.б. изменение поля `book_filename` в файле `_bookdown.yml` после возникновения ошибки.

2. Создавайте свою документацию

- Отредактируйте файл `index.Rmd` (содержит Вступление).
- Добавляйте собственные файлы, давая им именам по шаблону `<number>-<chapter-name>.Rmd`.
  Настоятельно рекомендуется ознакомиться с документацией на R Markdown и Kroki чтобы понимать правила, а так же можно опираться на примеры из проекта.
- Если у Вас уже имеется документация, описанная в формате markdown её можно добавить образом:

  - скопируйте файлы markdown в своб папку с документацией (назовём её условно `doc-folder`)
  - скопируйте изображения из документации в папку `docs_src/diagrams` или туда, где это покажется более логичным

  - измените markdown files to `.Rmd`
  - change names of markdown files so they would correspond to pattern `<number>-<chapter-name>.Rmd`
  - change references to local images in markdown files

- Если у Вас уже имеются текстовые описания диаграмм для графики в вашей документации и такой визуализатор поддерживается, то Вы можете включить их в документацию следующим образом:

  - скопируйте все необходимые файлы в папку `docs_src/diagrams` или туда, где это покажется более логичным
  - в файлах Rmd замените вставку изображений на вставку диаграмм так, как это описано здесь TODO

- Скорее всего Вам захочется использовать собственный класс Latex для создания PDF, поэтому добавьте файл `<your_latex_class>.cls` в вашу папку и укажите его в файле `index.Rmd` (замените поле `documentclass: book` названием вашего класса - `<your_latex_class>`).

  В случае если Вы будете использовать собственный класс Latex или свои пакеты для R, Python и т.д. - скорее всего потребуется включить дополнительные пакеты в docker образ `devdoc-swissknife`. В таком случае измените файл `docker/Dockerfile` как Вам требуется и соберите образ снова с помощью `make_docker.sh`.

Chapter 3

Примеры визуализации диаграмм с помощью Kroki

Примеры взяты с сайта Kroki, но я решил пропустить некоторые и оставить лишь те, которые на мой взгляд требуются чаще всего, а так же просты для текстового ввода.

Для полного перечня диаграмм, которые можно отобразить, стоит посмотреть документацию на поддерживаемые визуализаторы.

Данные для всех диаграмм в этой секции располагаются в файлах в папке docs_src/diagrams/examples данного репозитория.

Каждая диаграмма включена в документ с помощью следующей конструкции:

```
```{r echo=FALSE, results='asis'}
  to_diagram("from_src", "<Drawing name>", src="../diagrams/<src_file_path_within_diag
```
```

Такой шаблон применения описан в секции TODO

Код для всей секции с примерами можно увидеть здесь

3.1 C4 Диаграмма контекста (PlantUML+C4)

Визуализатор: `c4plantuml`

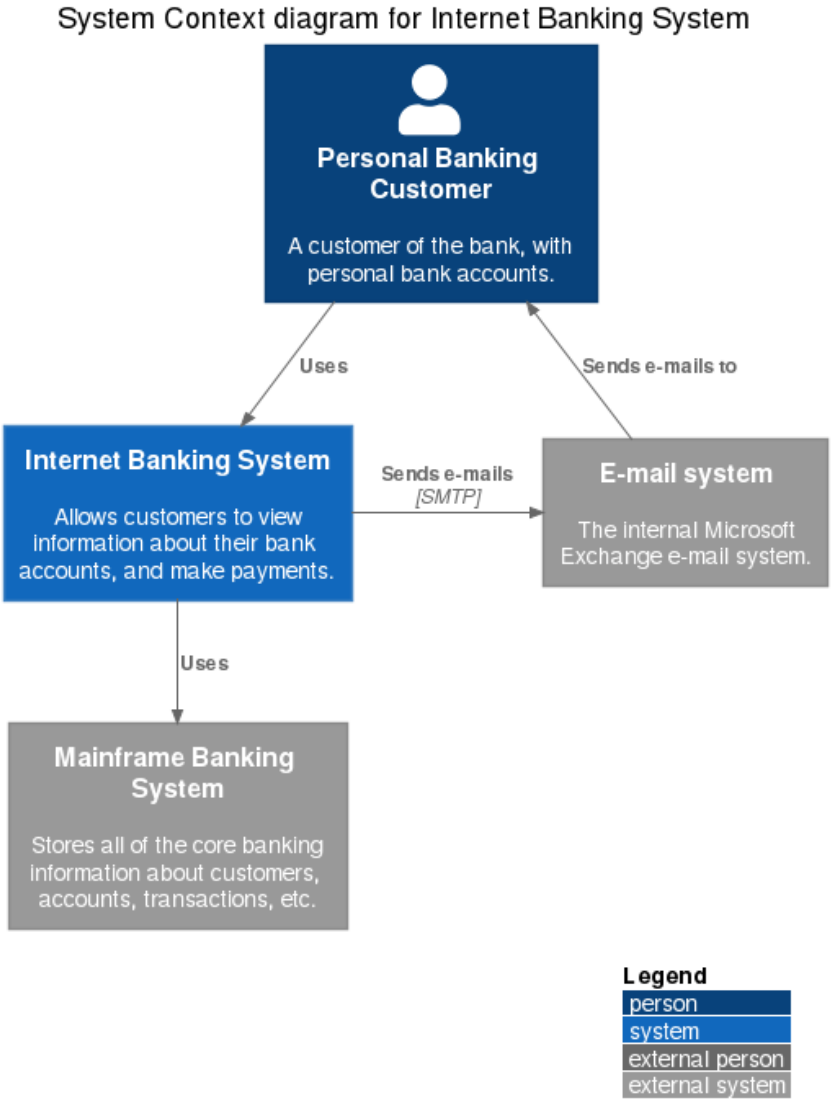

Figure 3.1: Example - C4 Context Diagram

*NOTE: Диаграммы c4plantuml приходится скачивать в виде PNG для документов PDF, т.к. при переводе этих SVG в PDF не всё проходит корректно

3.2 C4 Диаграмма контейнеров (PlantUML+C4)

Визуализатор: `c4plantuml`

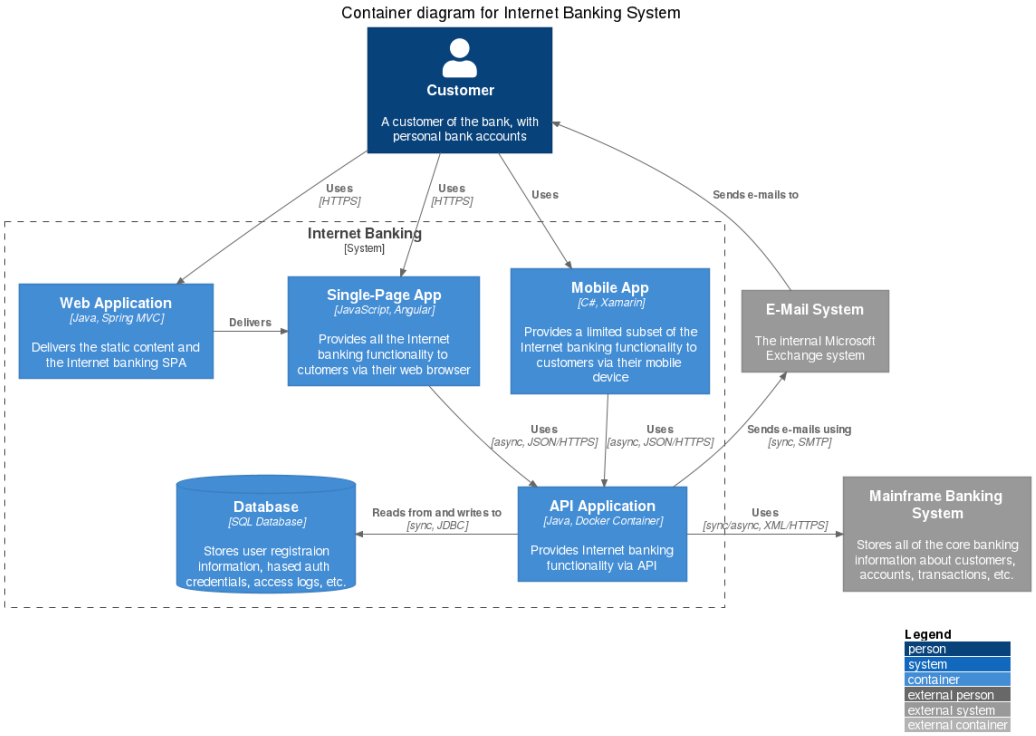

Figure 3.2: Example - C4 Container Diagram

*NOTE: Диаграммы c4plantuml приходится скачивать в виде PNG для документов PDF, т.к. при переводе этих SVG в PDF не всё проходит корректно

3.3 C4 Диаграмма компонентов (PlantUML+C4)

Визуализатор: `c4plantuml`

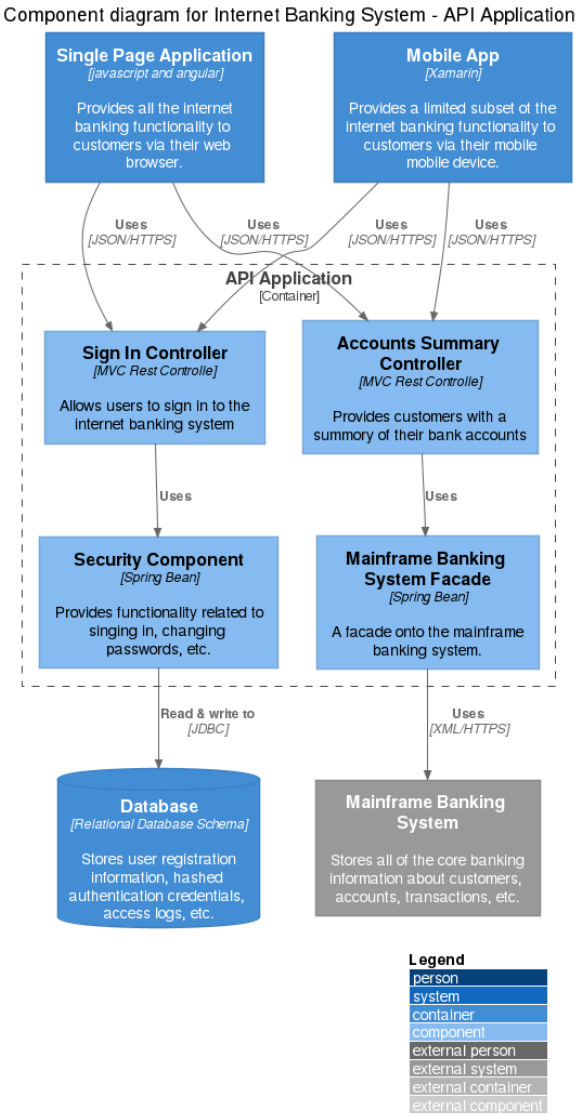

Figure 3.3: Example - C4 Component Diagram

*NOTE: Диаграммы c4plantuml приходится скачивать в виде PNG для документов PDF, т.к. при переводе этих SVG в PDF не всё проходит корректно

3.4 Блоксхема

Визуализатор: `blockdiag`

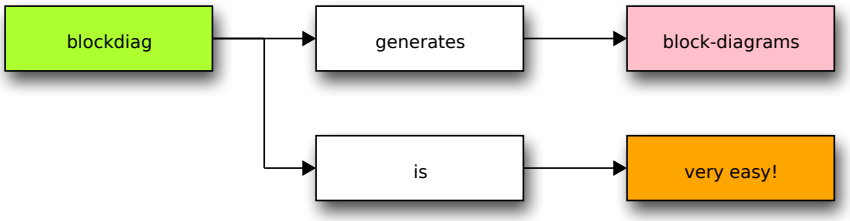

Figure 3.4: Example - Block Diagram

3.5 Цифровая временная диаграмма

Визуализатор: `wavedrom`

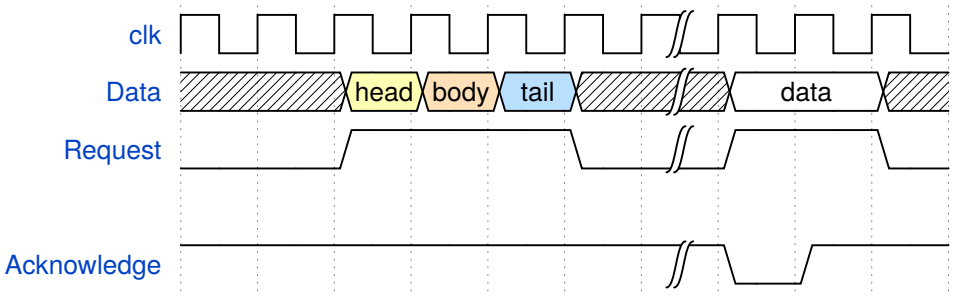

Figure 3.5: Example - Digital Timing Diagram

3.6 Bytefield

Визуализатор: `bytefield`

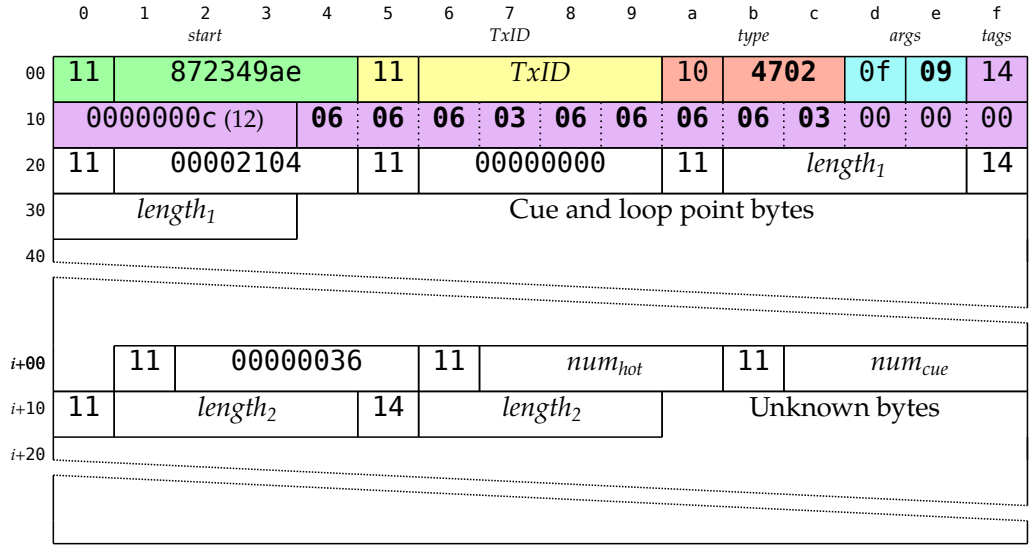

Figure 3.6: Example - Bytefield

3.7 Packet Diagram

Визуализатор: `packetdiag`

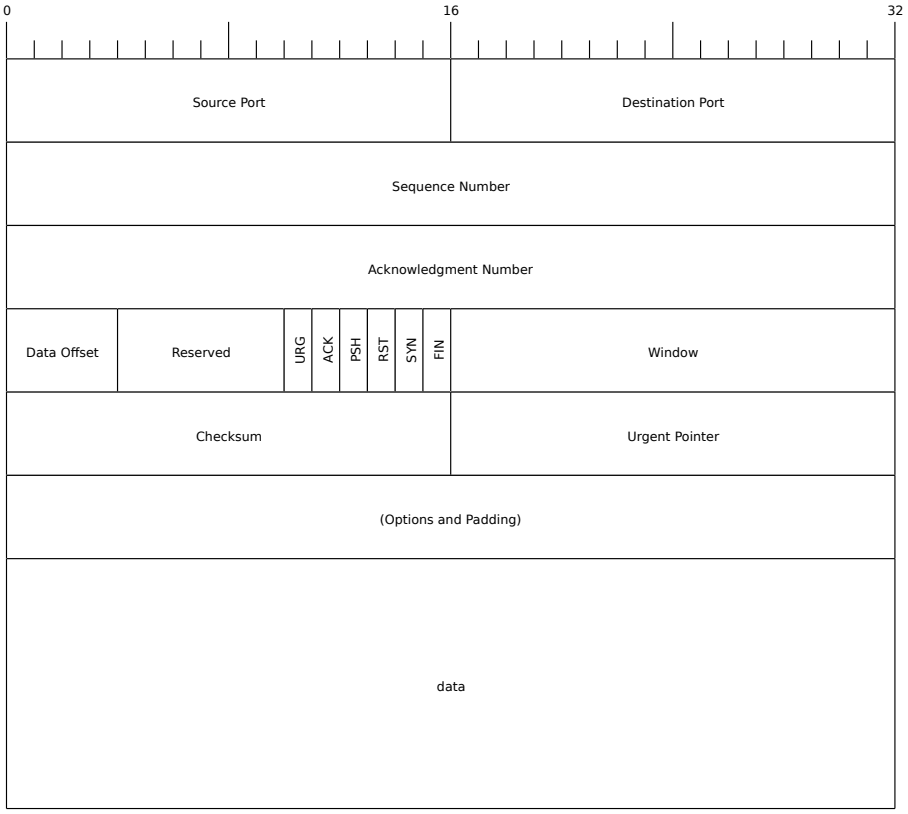

Figure 3.7: Example - Packet Diagram

3.8 Диаграмма последовательности №1 (PlantUML)

Визуализатор: `plantuml`

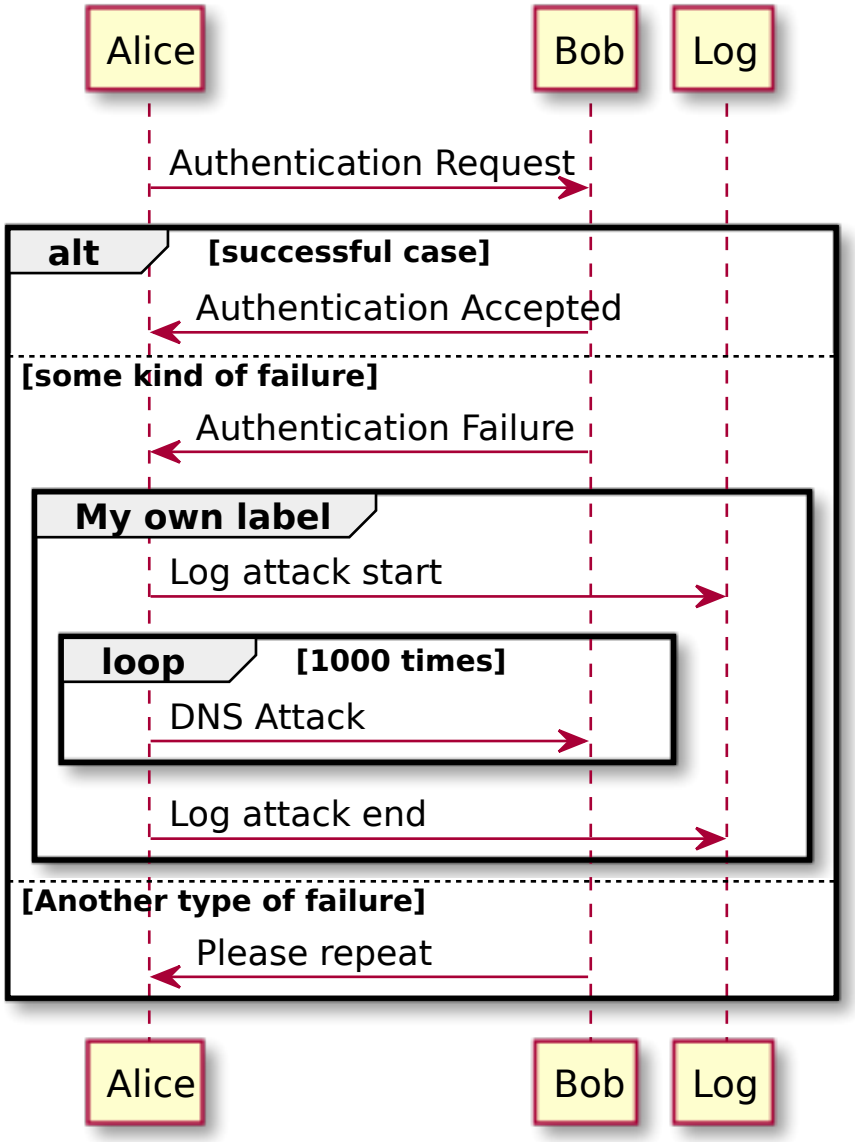

Figure 3.8: Example - Sequence Diagram - PlantUML

3.9 Диаграмма последовательности №2 (SeqDiag)

Визуализатор: `seqdiag`

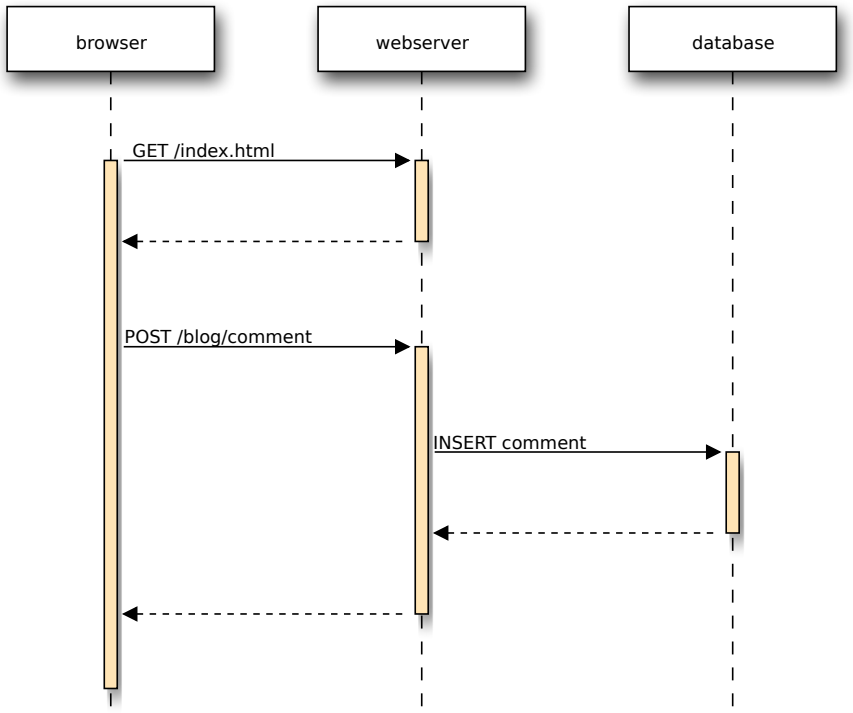

Figure 3.9: Example - Sequence Diagram - SeqDiag

3.10 Граф фиксации изменений

Визуализатор: `pikchr`

> NOTE: pikchr создаёт SVG, которые нормально отображают только Web-браузеры, поэтому с ними будут проблемы в PDF/PNG

3.11 Диаграмма прецедентов

Визуализатор: `plantuml`

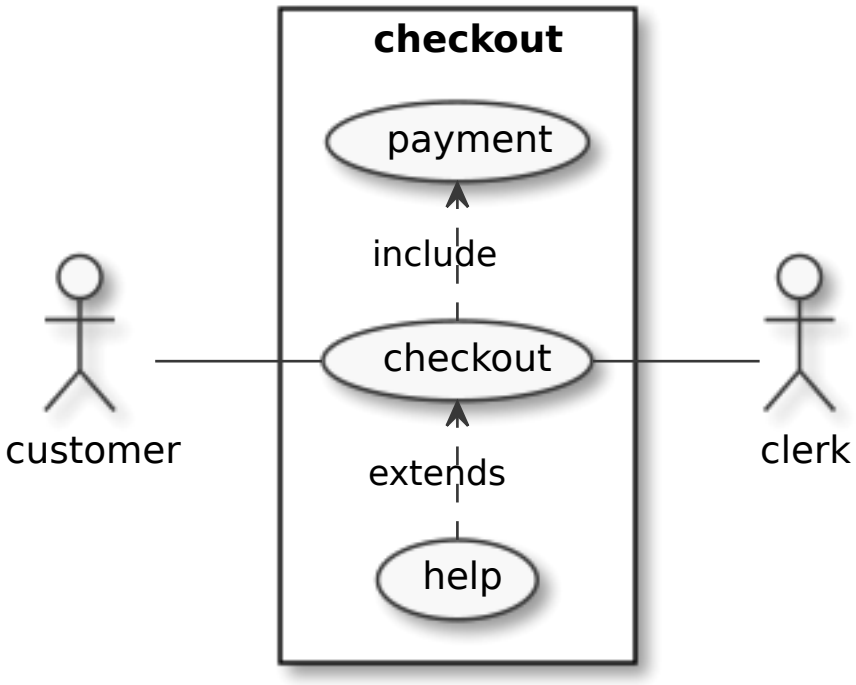

Figure 3.10: Example Block Diagram

3.12 Ментальная карта

Визуализатор: `plantuml`

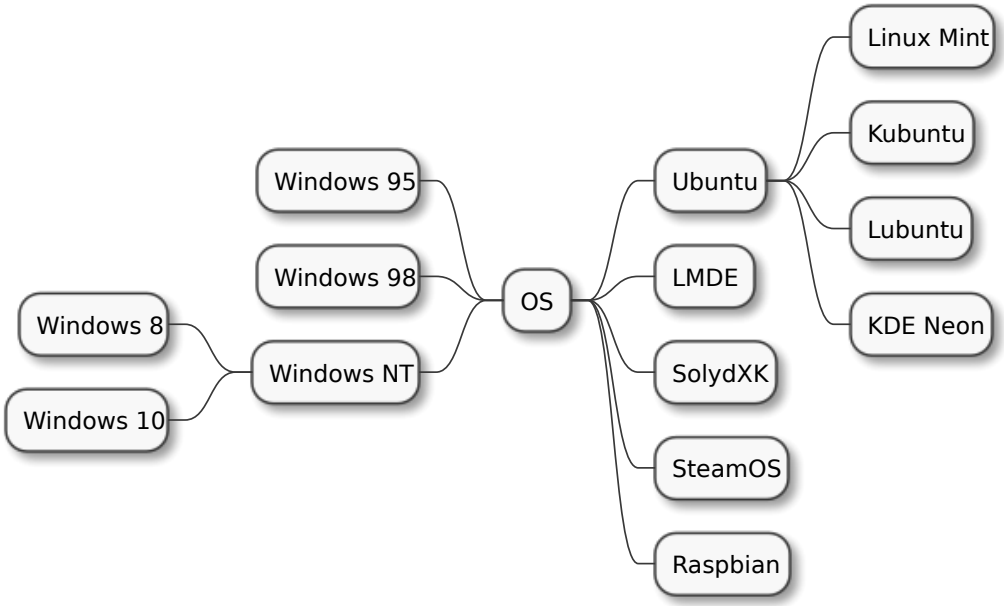

Figure 3.11: Example - Mind Map

3.13 PlantUML (ещё примеры)

В PlantUML возможно создание большого кол-ва диаграмм разного рода, таких как временные диаграммы, диаграмма Ганта и т.д..

Они все могут быть использованы аналогично предыдущим примерам.

Обратитесь к документации PlantUML https://plantuml.com/ для подробностей их описания.

3.14 Диаграмма Ганта

Визуализатор: `mermaid`

> NOTE: mermaid создаёт SVG, которые нормально отображают только Web-браузеры, поэтому с ними будут проблемы в PDF/PNG

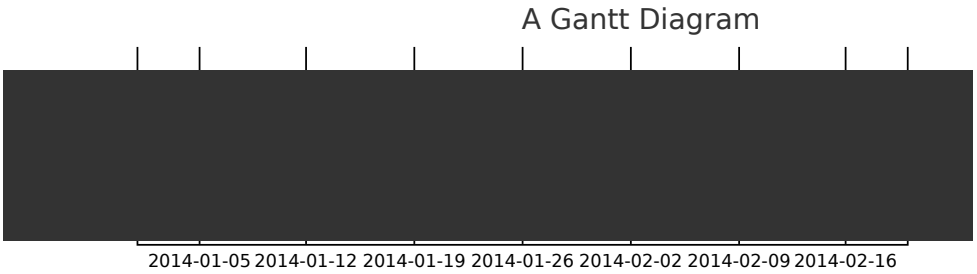

Figure 3.12: Example - Gantt