



Universidad de Oviedo



SOFTWARE
ARCHITECTURE

Software Architecture

Lab. 2

Advanced Github

Arc42

AsciiDoc

2019-20

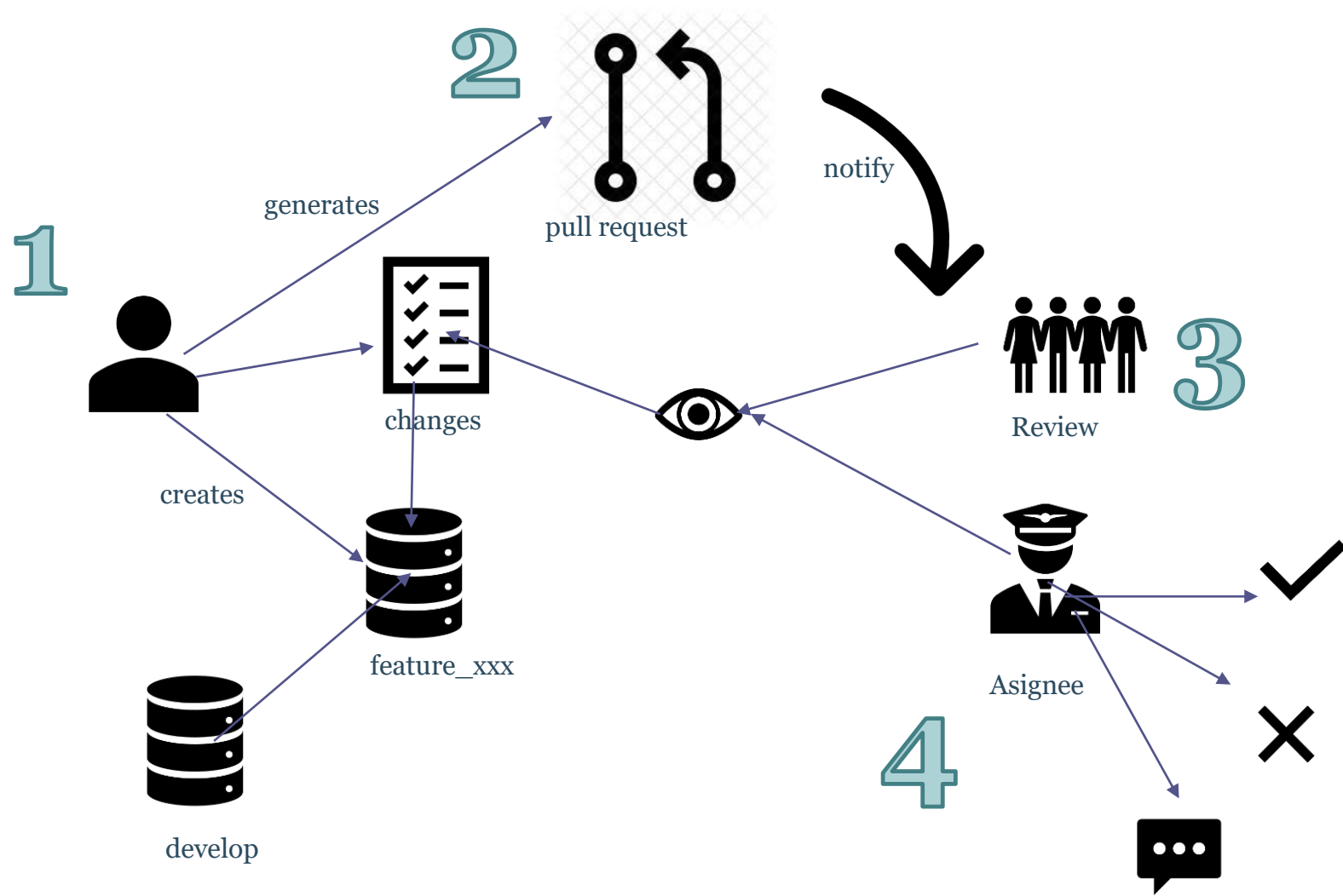
Jose Emilio Labra Gayo
Herminio García Gonz
Pablo González
Irene Cid

EN
English

Pull requests

- With **pull requests** we can tell others about the changes that you have made in a branch
- Others can **review** and **comment** the code before merging it to the base branch
- We have different roles: **coder**, **reviewer**, **assignee**

Pull request



Pull request - Steps

- Create the branch
 - `git checkout -b feature_1`
- Commit the changes
 - `git add .`
 - `git commit -m "changes in feature_1"`
- Push the changes
 - `git push --set-upstream origin feature_1`
- Go to GitHub and make a Pull request
- It is a **good alternative** to merging the changes by yourself.
- All the team can **participate** and **review** the code.

Your recently pushed branches:

🔗 **feature_1** (less than a minute ago)

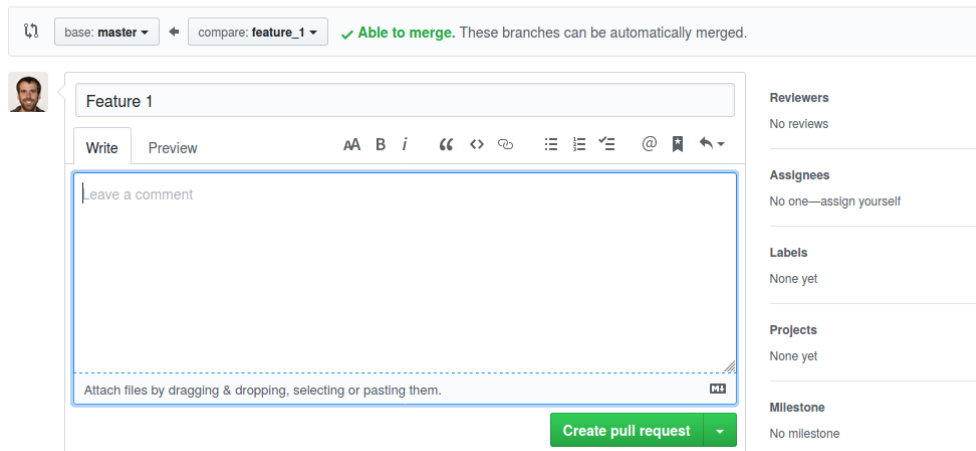
 [Compare & pull request](#)

Pull request - Steps

- Add a comment to describe the Pull request. Assign reviewers to it.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



The screenshot shows the GitHub interface for creating a pull request. At the top, there's a header with 'base: master' and 'compare: feature_1', followed by a green checkmark and the text 'Able to merge. These branches can be automatically merged.' Below this is a section for 'Feature 1' with a 'Write' tab selected. A large text area for 'Leave a comment' is highlighted with a blue border. To the right of the text area is a sidebar with sections for 'Reviewers', 'Assignees', 'Labels', 'Projects', and 'Milestone', each with a gear icon. At the bottom right of the text area is a green button labeled 'Create pull request'.

- Reviewers have three different options: accept the changes, add comments or reject the changes

Arc42



Template for creating documentation for software projects



Simple, fast and clear. Open source and free of charge.



Available in multiple formats:

- **Asciiidoc**
- Word (docx) Markdown
- LaTeX ..

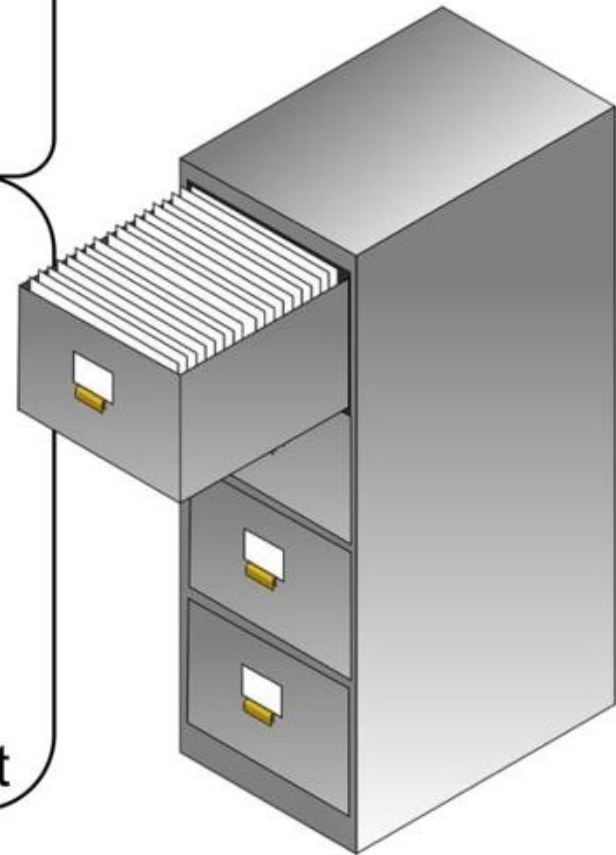
Arc42

Problem

- 1.- Introduction and goals
- 2.- Constraints
- 3.- Context & scope

Solution

- 4.- Solution strategy
- 5.- Building block view
- 6.- Runtime view
- 7.- Deployment view
- 8.- Crosscutting concepts
- 9.- Architectural decisions
- 10.- Quality requirements
- 11.- Risks and technical debt
- 12.-Glossary



1 - Introduction and goals

- Short description of the **requirements**, driving forces, extract (or abstract) of requirements.
- Top three (max five) **quality goals** for the architecture which have highest priority for the major stakeholders.
- A table of important **stakeholders** with their expectation regarding architecture.



2 - Constraints

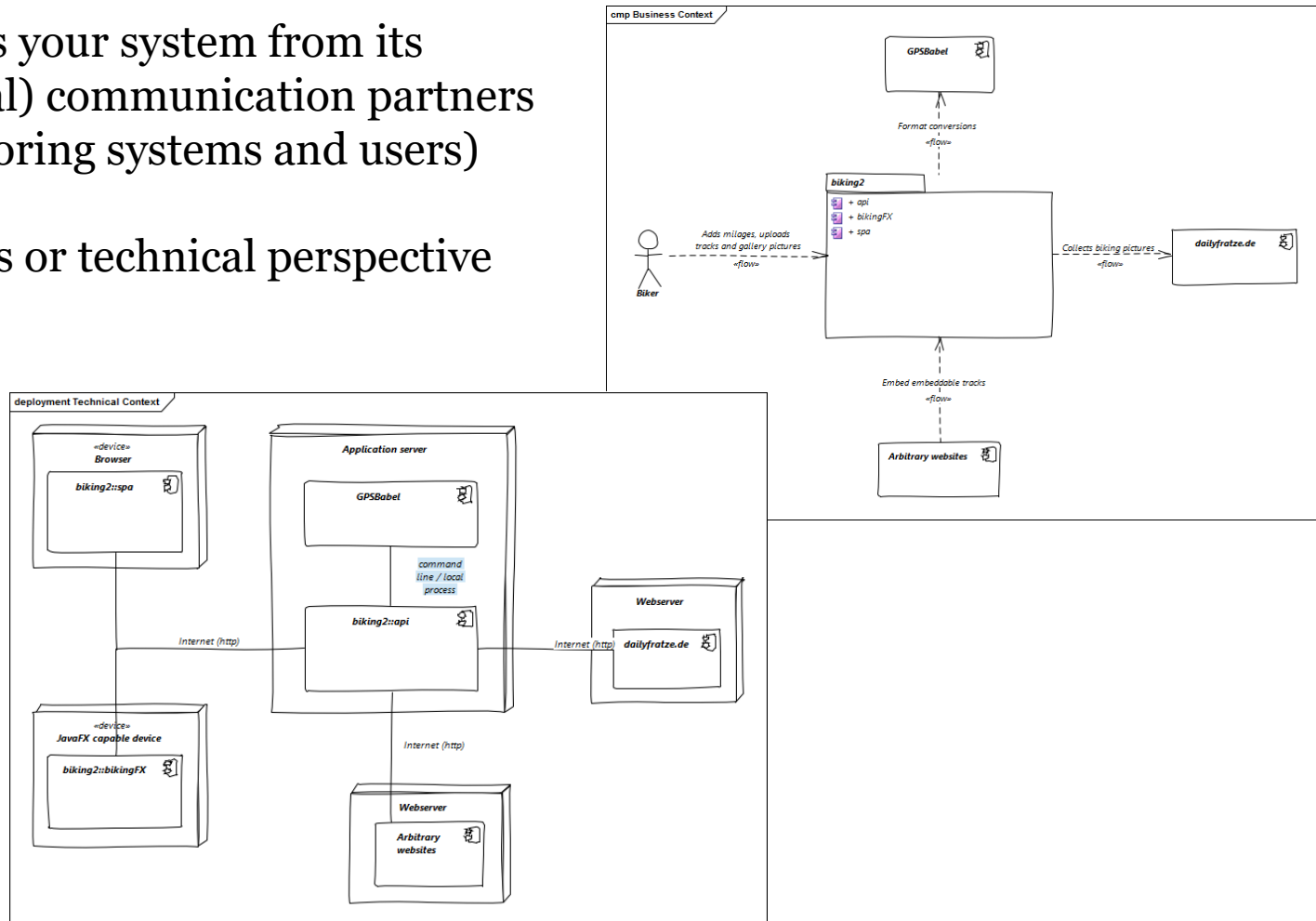
- Anything that constrains teams in design and implementation decisions or decision about related processes.
- Can sometimes go beyond individual systems and are valid for whole organizations and companies.
- Technical constraints, but also time, money, etc.



Constraint	Explanation
...	...

3-Context and scope

- Delimits your system from its (external) communication partners (neighboring systems and users)
- Business or technical perspective



Templates - AsciiDoc

```
01_introduction_and_goals.adoc
[[section-introduction-and-goals]]
== Introduction and Goals

[role="arc42help"]
****
Describes the relevant requirements and the driving forces that software architects and
development team must consider. These include

* underlying business goals, essential features and functional requirements for the system
* quality goals for the architecture
* relevant stakeholders and their expectations
****

=== Requirements Overview

[role="arc42help"]
****
.Contents
Short description of the functional requirements, driving forces, extract (or abstract) of requirements. Link to (hopefully existing) requirements documents (with version number and information where to find it).

.Motivation
From the point of view of the end users a system is needed or
improve support of a business activity and/or improve

.Form
Short textual description, probably in tabular use-case format.
If requirements documents exist this overview should refer to them
```

ASCIIDOCTOR

Introduction and Goals

Describes the relevant requirements and the driving forces that software architects and development team must consider. These include

- underlying business goals, essential features and functional requirements for the system
- quality goals for the architecture
- relevant stakeholders and their expectations

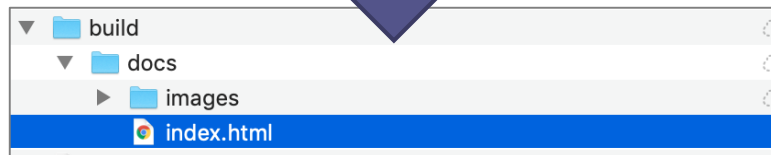
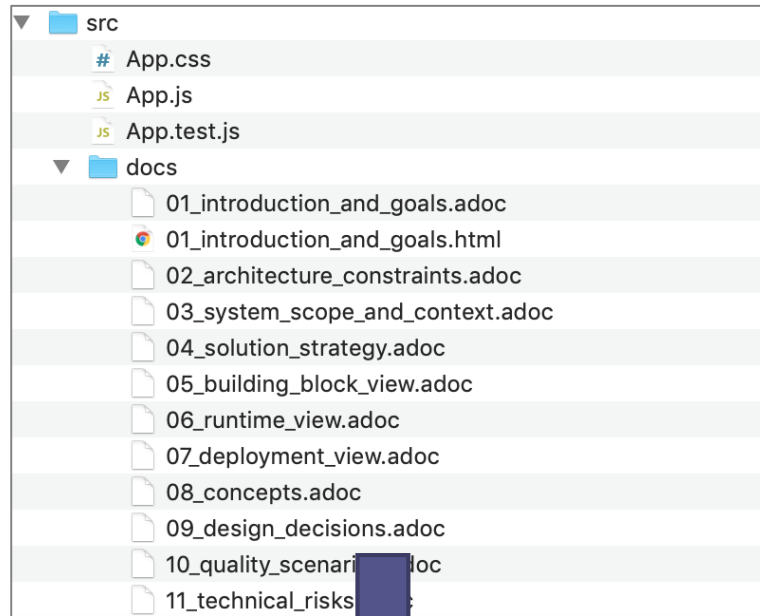
Requirements Overview

Contents

Short description of the functional requirements, driving forces, extract (or abstract) of requirements. Link to (hopefully existing) requirements documents (with version number and information where to find it).

Motivation

Documentation in Viade



Everything is automated with npm:
`$ npm run docs`

This will execute:

```
$ asciidoctor -D build/docs -a  
imagesdir=./images -r asciidoctor-  
diagram src/docs/index.adoc && cp -R  
src/docs/images build/docs
```

We need asciidoctor and asciidoctor-
diagram installed:

```
$ gem install asciidoctor asciidoctor-  
diagram
```

Lab 02. - Let's practise

- Groups will be divided in 3 groups of 2/3 people each.
- Each group will create a branch in order to work in one part of the documentation:
 - `feature_arc42_01introduction`
 - `feature_arc42_02contraints`
 - `feature_arc42_03context`
- When the group finishes the work, make a Pull Request.
- The other developers should review this Pull Request before merging them into the master branch.
- Everybody contributes in writing the documentation!
- Do not forget to write the minutes!!

If you need help with asciidoc, check this [page](#)