University of Applied Sciences Rapperswil
Department of Computer Science

Bachelor Thesis

XMPP-Grid Broker

Authors:
Fabian Hauser and
Raphael Zimmermann

Advisor:
Prof. Dr. Andreas Steffen

External Co-Examiner:
Dr. Ralf Hauser
PrivaSphere AG

Internal Co-Examiner:
not yet defined

Spring Term 2018

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

DON‘T PANIC

“It looked insanely complicated, and this was one of the reasons why the snug plastic cover it fitted into had the words DON‘T PANIC printed on it in large friendly letters.”

The Hitchhiker‘s Guide to the Galaxy

Abstract

Fabian HAUSER and Raphael ZIMMERMANN

XMPP-Grid Broker

Management Summary

Acknowledgements

We would like to thank our advisor, Prof. Dr. Andreas Steffen, for his continuous support and helpful comments.

Contents

Chapter 1

Introduction

In this chapter, we introduce the terminology and background of our thesis, to conclude in the motivation and legitimisation of our thesis.

1.1 Terminology

Taking into account that the intended audience for this thesis are developers and operators of security reporting systems, we mostly use the Security Automation and Continuous Monitoring (SACM) terminology [2] and thereby follow the same guidelines as the XMPP grid draft [4].

1.2 Background

The following sections introduce the underlying XMPP protocol and the relevant extensions (*XEPs*) used by the XMPP grid draft as well as a summary of it and the corresponding XMPP terminology.

1.2.1 XMPP (eXtensible Messaging and Presence Protocol)

The Extensible Messaging and Presence Protocol (in short *XMPP*) is an open protocol that enables the near-real-time exchange of small data between any network endpoints, hereafter called *Platforms* [12]. While originally designed as an Instant Messaging (IM) protocol, XMPP can be used for a wide range of data exchange applications [11].

XMPP is made of small building blocks defined in the core protocol [12] and numerous extensions called *XEPs* [13]. The core is comprised of functionality for setup and encryption of communication channels, *XML* streams, error handling and more. Additional functionality such as *Service Discovery* [7] and *Publish-Subscribe* [8] are defined in separate extensions.

Although XMPP supports peer-to-peer communication, it is often used in a traditional client-server architecture. A client (*Platform*) can send data to any addressable entity (any other *Platforms*) using *Jabber* Identifiers, hereafter called *JID*. If the *JID* of the receiver has a different domainpart than the current server (*Controller*), the message is forwarded to the responsible XMPP server under its domain [12].

The data exchanged over XMPP is *XML* which makes the protocol structured and extensible, but leads to some protocol overhead. XMPP communicates over unidirectional data streams with a server, which are basically long-lived *TCP* connections. The client opens a channel to the server over this connection, and the server opens one back (i.e. `<stream>` XML tags). In both streams, an XML document is opened after the connection is established. During the conversation, an arbitrary amount of *stanzas* (specified XML child elements) are written to the stream. Before a connection may be terminated, the root element is closed (i.e. `</stream>`) and both streams form valid XML documents [12][9].

The core *stanza* types are *Messages* (`<message/>`), *Presence* (`<presence/>`) and *Info/Query* (`<iq/>`). *Messages* can contain arbitrary data similar to email but are optimised for immediate delivery. *Presence stanzas* deal with network availability and the propagation of user presence information. An *Info/Query stanza* consists of a request and response (similar to the GET and POST HTTP methods), which is used for feature negotiation, configuration and general information exchange. Because of these coarse semantics, XMPP provides a generalized communication layer [12][11].

Figure 1.1 illustrates an example setup with two servers and three clients.

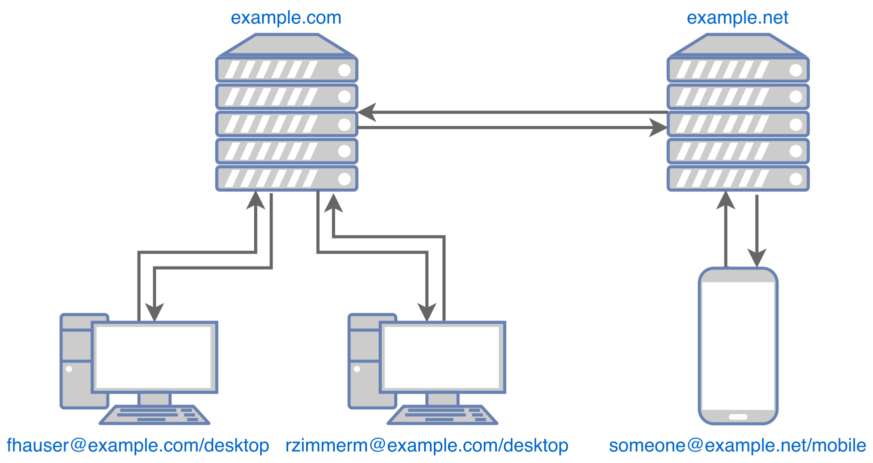

example.com
example.net
fhauser@example.com/desktop
rzimmerm@example.com/desktop
someone@example.net/mobile

FIGURE 1.1: Two XMPP domains (servers), one with two users and one with a single mobile user.

1.2.2 Relevant XMPP Extensions

The XMPP grid draft [4] is based on multiple *XEPs*, most notably *Publish-Subscribe*. In this section, we give an overview of the most relevant used *XEPs*.

XEP-0004: *Data Forms* is a flexible protocol that can be used in workflows such as service configuration as well as for application-specific data description and reporting. The protocol provides form processing, common field types and extensibility mechanisms [5].

XEP-0030: *Service Discovery* enables entities to discover information about the identity and capabilities of other entities, e.g. whether the entity is a server or not, or items associated with an entity, e.g. a list of *Publish-Subscribe* nodes [7].

XEP-0059: *Result Set Management* allows entities to manage the receipt of large result sets, e.g. by paging through the result or limiting the number of results. *Result Set Management* is often desired when dealing with large dynamic result sets, as from service discovery or publish-subscribe, and time or other resources are limited [10].

XEP-0060: *Publish-Subscribe*

The *Publish-Subscribe* Extension, hereafter referred to as *PubSub* or *Broker*, enables XMPP entities (*Provider*) to broadcast information via *Topics* to subscribed entities (*Consumer*) [8].

Nodes, hereafter referred to as *Topics*, are the communication hubs. Entities can create topics and configure them, e.g. set up subscription timeouts or limit publishing and subscription rights. The configuration mechanism is based on data forms (XEP-0004). An XMPP-Server ***may*** restrict node creation to certain entities, which means that possibly not every XMPP-Server that supports *Publish-Subscribe* also implements this feature [3].

The protocol defines a hierarchy of six affiliations of which only the implementation of 'owner' and 'none' is ***required***. The implementation of the remaining four affiliations is ***recommended***. An owner of a topic can manage the subscriptions and affiliations of other entities associated with a given topic.

To simplify the creation of topics, *PubSub* defines five topic access models ("node access models") that ***should*** be available: open, presence, roaster, authorize and whitelist.

The open model allows uncontrolled access while presence and roaster are specific for IM. Using the authorize model, the owner has to approve all subscription requests. The whitelist model enables the owner to maintain a list of entities that are allowed to subscribe.

1.2.3 IETF Internet-Draft: Using XMPP for Security Information Exchange

This IETF Internet draft describes how the XMPP protocol enhanced with the previously discussed XEPs, most notably the *PubSub* extension, can be used for the exchange and distribution of security-relevant information between network devices.

One of the primary motivation for using XMPP for this task is the fast propagation of such security-relevant data. Using XMPP for such a task also comes with its downsides. Most notably, because the XMPP server (*Broker*/*Controller*) is the central configuration component in charge of managing access permission, its compromisation has serious consequences.

The draft describes a trust model, thread model as well as specific countermeasures, e.g. to use at least TLS 1.2. These countermeasures also define restrictions of the XMPP protocol and its extensions, e.g. by limiting the topic access models of *PubSub* to whitelist and authorized only [4].

1.3 Motivation

In this section, we legitimate this thesis and explain the value and applicability of our proposed solution.

1.3.1 Present situation

The IETF standard draft ***Using XMPP for Security Information Exchange*** [4] as summarised in section 1.2.3 defines a protocol to exchange security-relevant information between endpoints. The draft was created by the Managed Incident Lightweight Exchange (MILE) Working Group to support computer and network security incident management.

To demonstrate the viability of the draft a rapid prototype was developed in November 2017 [14].

1.3.2 Problem and Solution

Currently, there exists no implementation of the *XMPP* grid draft management functionality which is ready for production use regarding usability and security.

To solve this problem, a graphical interface with bindings to a suitable *Broker* must be proposed and implemented. The interface should permit network administrators to manage and review *Topics*, persisted items and *Platforms*. Additionally, subscription and publishing permissions of *Topics* and *Platforms* must be manageable.

Bibliography

[1] Agile Alliance. What are user stories? https://www.agilealliance.org/glossary/user-stories, 2015.

[2] H. Birkholz, J. Lu, J. Strassner, N. Cam-Winget, and A. W. Montville. Security Automation and Continuous Monitoring (SACM) Terminology. Internet-Draft draft-ietf-sacm-terminology-14, Internet Engineering Task Force, Dec. 2017. Work in Progress.

[3] S. O. Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, Mar. 1997.

[4] N. Cam-Winget, S. Appala, S. Pope, and P. Saint-Andre. Using XMPP for Security Information Exchange. Internet-Draft draft-ietf-mile-xmpp-grid-05, Internet Engineering Task Force, Feb. 2018. Work in Progress.

[5] R. Eatmon, J. Hildebrand, J. Miller, T. Muldowney, and P. Saint-Andre. Data Forms. XEP-0004, Aug. 2007.

[6] S. Hares, J. Strassner, D. Lopez, L. Xia, and H. Birkholz. Interface to Network Security Functions (I2NSF) Terminology. Internet-Draft draft-ietf-i2nsf-terminology-05, Internet Engineering Task Force, Jan. 2018. Work in Progress.

[7] J. Hildebrand, P. Millard, R. Eatmon, and P. Saint-Andre. Service Discovery. XEP-0030, Oct. 2017.

[8] P. Millard, P. Saint-Andre, and R. Meijer. Publish-Subscribe. XEP-0060, Feb. 2018.

[9] J. Moffitt. *Professional XMPP Programming with JavaScript and jQuery*. Wrox Press Ltd., Birmingham, UK, UK, 2010.

[10] I. Paterson, P. Saint-Andre, V. Mercier, and J.-L. Seguineau. Result Set Management. XEP-0059, Sept. 2006.

[11] P. Saint-Andre. Streaming xml with jabber/xmpp. *IEEE Internet Computing*, 9(5):82–89, Sept 2005.

[12] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120, Mar. 2011.

[13] P. Saint-Andre and D. Cridland. XMPP Extension Protocols. XEP-0001, Nov. 2016.

[14] A. Steffen. XMPP-Grid HSR Rapid-Prototype. https://github.com/sacmwg/vulnerability-scenario/blob/b3bf6a5b21f242b788488ba8991595172fe663e7/ietf_hackathon/strongSwan/pubsub_client.py, Nov 2017.

List of Figures

List of Tables

Glossary

Broker

"A *SACM* Broker Controller is a *Controller* that contains *control plane* functions to provide and/or connect services on behalf of other *SACM* components via interfaces on the *control plane*" [2]

Component

An encapsulation of software that communicates using Interfaces, that is composed of *SACM* capabilities. [2, 6]

Consumer

"In *SACM*, an entity that contains functions to receive information from other *Components;* as used here, the term refers to an *XMPP Publish-Subscribe* Subscriber." [4]

control plane

"An architectural component that provides common control functions to all *SACM* components." [2]

Controller

"In SACM, a ‘component containing control plane functions that manage and facilitate information sharing or execute on security functions‘; as used here, the term refers to an XMPP server, which provides core message delivery [RFC6120] used by publish-subscribe entities." [4, 2]

Data Forms

XMPP Data Forms Extension

Info/Query

XMPP Info/Query *stanza*

Jabber

Originial Name of *XMPP*

JID

Jabber IDentifier, e.g. bob@example.com/mobile

Message

XMPP Message *stanza*

Platform

"Any entity that connects to the *XMPP*-Grid in order to publish or consume security-related data." [4]

Presence

XMPP Presence *stanza*

Provider

"In *SACM* An entity that contains functions to provide information to other *Components*; as used here, the term refers to an *XMPP Publish-Subscribe* Publisher." [4]

Publish-Subscribe

XMPP Publish Subscribe Extension

PubSub

Common abbreviation for *Publish-Subscribe*

Result Set Management

XMPP Result set Management Extension

SACM

Abbreviation for Security Automation and Continuous Monitoring

Service Discovery

XMPP Service Discovery Extension

stanza

A *XMPP* a fragment of XML that is sent over a stream, e.g. *Message*.

TCP

Transmission Control Protocol

TLS

Transport Layer Security

Topic

"A contextual information channel created on a Broker at which messages generated by a Provider are propagated in real time to one or more Consumers. Each Topic is limited to a specific type and format of security data (e.g., IODEF) and provides an XMPP interface by which the data can be obtained." [4]

XEP

XMPP Extension Protocol

XML

eXtensible Markup Language

XMPP

eXtensible Messaging and Presence Protocol

Appendices

A.1 Task Description

TODO

A.2 Project Plan

XMPP-Grid Broker: Project Plan

Authors:
Fabian HAUSER and
Raphael ZIMMERMANN

Advisor:
Prof. Dr. Andreas STEFFEN

Spring Term 2018

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Contents

Chapter 1

Project Overview

The goal of the bachelor thesis is to build a broker application and graphical user interface to administer XMPP-Grids according to draft-ietf-mile-xmpp-grid, as described in the task description [3].

Chapter 2

Project Organization

All team members have the same strategic rights and duties. Prof. Dr. Andreas Steffen is our project advisor.

2.1 Roles

Due to the small team size, most roles are performed by both team members.

Raphael Zimmermann
project management, software engineering, quality assurance.

Fabian Hauser
infrastructure management, software engineering, quality assurance.

Chapter 3

Project Management

3.1 Components

For a better overview and to allow us a sophisticated time assessment, we decided to group tasks into categories, i.e. JIRA components. Components represent processes, documents and products which are to be released.

Currently, tasks are separated into following components:

- Application
- Final Submission Document
- Management
- Poster
- Presentation
- Project Plan

3.2 Time Budget

The project started with the Kickoff Meeting on 19.02.2018 and will be completed after 16 weeks by 15.06.2018. The two team members are available for 360 hours each during the semester which corresponds to a weekly time budget of 20 hours per person and two weeks with a weekly time budget of 40 hours per person.

Apart from the statutory holidays, there are no further absences planned.

3.3 Schedule

The project schedule is an iterative process based on elements of SCRUM.

We decided on a sprint duration of approximately week, but allow deviations in working hours depending on statutory holidays.

3.3.1 Iterations & Milestones

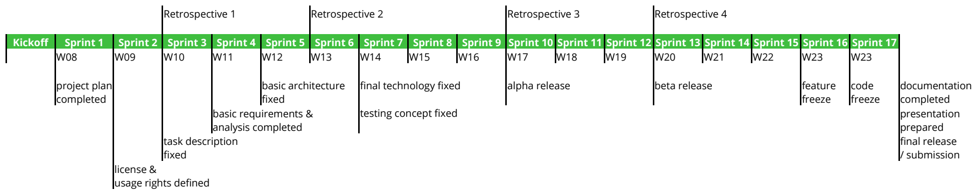


FIGURE 3.1: Overview of the 16 iterations

The goals and milestones resulting from each sprint are shown in Figure 3.2.

TABLE 3.1: Meeting Time Budget

Meeting Type	**Total Duration per Person**	**Total Duration for the Team**
Supervision Meetings	12 hours	24 hours
Standup Meetings	16 hours	32 hours
Sprint Planning Meetings	18 hours	36 hours
Retrospective	4 hours	8 hours
Total	50 hours	100 hours

3.3.2 Meetings

The team works every Monday (08:00 - 12:00) and Tuesday (08:00 - 17:00) together in the study room and remotely Fridays (08:00 - 17:00). Wednesday and Friday begin with a daily stand-up meeting taking no longer than 15 minutes. Sprint planning meetings are carried out on Tuesday at 10:00. Table 3.1 shows an overview of the total meeting time budget.

Regular meetings with the project advisor take usually place on Monday in Prof. Dr. Steffen's office.

Raphael Zimmermann will take meeting notes for every meeting. Meeting minutes are published on the project website afterwards.

Sprint	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Tag	**0.1.0**	**0.2.0**	**0.3.0**	**0.4.0**
Date	20.02.2018 – 27.02.2018	27.02.2018 - 06.03.2018	06.03.2018 – 13.03.2018	13.03.2018 – 20.03.2018
Milestones	- license & usage rights defined	- task description fixed	- basic requirements & analysis completed	- basic architecture fixed
Time Budget	40	40	40	40
Tasks	- setup remaining project infrastructure - read ietf draft, XEP standards etc. - compile list of open questions - begin with chapter "initial situation" - analyze existing python proof-of-concept	- collect non-functional requirements (NFRs) - compile set of user stories - draw wireframes - research frameworks and technology: Prepare architectural decisions	- extend NFRs, user stories, wireframes - make first architectural decisions - further framework and technology research - implement proof of concepts for ciritical components	- make big architectural decisions - implement proof of concepts for ciritical components
Documents / Chapters / Artefacts		- chapter "Initial Situation" completed - signed task description		

Sprint	Sprint 5	Sprint 6	Sprint 7	Sprint 8
Tag	**0.5.0**	**0.6.0**	**0.7.0**	**0.8.0**
Date	20.03.2018 – 27.03.2018	27.03.2018 – 03.04.2018	03.04.2018 – 10.04.2018	10.04.2018 – 17.04.2018
Milestones		- final technology fixed - testing concept fixed		
Time Budget	40	40	40	40
Tasks	- implement proof of concepts for ciritical components - draft testing concept	- implement proof of concepts for ciritical components - finalise testing concept	- Implement most important user stories - make further architectural decisions	- Implement most important user stories - make further architectural decisions
Documents / Artefacts		- Chapter "Our Approach" completed		

Sprint	Sprint 9	Sprint 10	Sprint 11	Sprint 12
Tag	**0.9.0**	**0.10.0**	**0.11.0**	**0.12.0**
Date	17.04.2018 – 24.04.2018	24.04.2018 – 01.05.2018	01.05.2018 – 08.05.2018	08.05.2018 – 15.05.2018
Milestones	- alpha release			- beta release
Time Budget	40	40	40	40
Tasks	- Implement most important user stories - make further architectural decisions - write integration tests	- Implement secondary user stories - write integration tests	- Implement secondary user stories - write integration tests	- Implement secondary user stories - write integration tests
Documents / Artefacts	- alpha release bundle			- beta release bundle

Sprint	Sprint 13	Sprint 14	Sprint 15	Sprint 16
Tag	**0.13.0**	**0.14.0**	**0.15.0**	**0.16.0**
Date	15.05.2018 – 22.05.2018	22.05.2018 – 29.05.2018	29.05.2018 – 05.06.2018	05.06.2018 – 12.06.2018
Milestones			- future freeze	- code freeze
Time Budget	40	40	40	80
Tasks	- Implement remainging user stories - Improve code base - write integration tests	- Implement remainging user stories - Improve code base - write integration tests	- Implement remainging user stories - Improve code base - write integration tests	- fix eventual bugs - write documentation
Documents / Artefacts				- final release bundle - poster - abstract

Sprint	Sprint 17
Tag	**1.0.0**
Date	12.06.2018 – 15.06.2018
Milestones	- documentation completed - presentation prepared - final release / submission
Time Budget	40
Tasks	- write documentation - submit documents
Documents / Artefacts	- personal reports - management summary - presentations - time accounting

FIGURE 3.2: Detailed overview with tasks and milestones of all 16 sprints.

Chapter 4

Risk Management

An assessment of the project-specific risks is carried out in Table 4.2 as time loss during the whole project. The risk matrix in Table 4.1 provides an overview of the risk weighting.

To account for these risks, we reduce our weekly sprint time by the total weighted risk applicable to the planned task topics (on average approximately 13.5%). We also review the risk assessment after every sprint, adapt it and take measures if necessary.

Severity / Probability	High ($\geq 5d$)	Medium (2-5d)	Low ($\leq 2d$)
High ($\geq$ 60%)	1		
Medium (30-60%)	6		
Low ($\leq$ 30%)		3, 4, 5	

TABLE 4.1: The risk matrix. Numbers reference to the risk assessment Table 4.2

TABLE 4.2: Risk assessment table. Time in hours over the total project duration.

#	Title	Description	Prevention / Reaction	Risk [h]	Probability	= [h]
1	Incomplete reference documentation	The reference documentation / standards are incomplete or difficult to comprehend.	Discuss missing parts with project advisor	60	60%	36
2	Communication errors	Errors due to miscommunication or misapprehension.	Maintain a high level of interaction, precise specification of tasks responsibilities, conduct meetings if ambiguities exist.	30	50%	15
3	Problems with project infrastructure	The used project infrastructure is not or only partially available, or data loss occurs within management software.	Clean setup and self-hosting of the tools to prevent third-party dependencies.	45	30%	13.5
4	Scope creep	The project's scope is extended over the project course.	Define the project scope and limitations precisely. Discuss changes with the project advisor.	45	30%	13.5
5	Dependency errors	There are errors/bugs in third-party dependencies, i.e. libraries.	Carefully select libraries and limit thid-party dependency to a minimum.	30	30%	9
6	Missing dependency documentation	Selected libraries are lacking proper documentation	The documentation quality of a library should be a selection criterion.	30	40%	12
	Total weighted risk					99

Chapter 5

Infrastructure

5.1 Project Management and Development

For project management we utilise JIRA[1]. As document/code storage, git repositories on Github are used, the continuous integration/deployment will be defined in the course of the project.

These applications are hosted on our HSR project server, which runs a standard Ubuntu Linux 17.10.

5.1.1 Development Tools

The development tools will be defined in the course of the project.

5.2 Backup and Data Safety

An incremental backup of the project server including the source code and documentation is created on an independent system every night.

As our documents and code is stored in a git repository, they are also distributed on all development systems.

Chapter 6

Quality Measures

To maintain a high standard of quality, we take the following measures:

- short sprint reviews
- four extended retrospectives
- code reviews
- automated unit and integration testing
- publish all documentation on the project website using continuous integration/delivery.
- using continuous integration for source code

6.1 Documentation

The official documents such as the final submission document, meeting minutes as well as this project plan are written in LaTeX respectively ASCIIDoc and published on the project website [1] containing all PDF documents.

The sources are in both cases kept under version control in the same repository, which allows us to use the same tools and processes for documentation and code. The continuous integration server builds and publishes the website whenever new changes are pushed to the repository.

6.2 Project Management

Because the project plan allows for an iterative process, we use JIRA with its SCRUM-features (such as sprint creation or boards) for project management.

6.2.1 Sprint Planning

Each sprint is mapped to JIRA, which allows the project advisor to trace the project progress. Sprints are represented as boards on which the current state and assignee of any issue is easily visible ("To Do", "In Progress", "Review", "Done").

6.2.2 Definition of Done

An issue may be closed if ***all*** of the following conditions are met:

- All functionality conforms to the specification. Any deviations must be discussed and decided by the team.
- The source code is reasonably documented.

[1] https://xgb.redbackup.org

- No code is commented out.
- No warnings and errors by the compiler or any other quality tool.
- A review is performed and accepted in a pull request.
- The corresponding branch is merged into the stable branch (e.g. master).
- All documents are up to date including the project website.
- Reasonable unit and integration tests exist and pass.
- The complete continuous integration pipeline works.
- All time is logged.

6.3 Development

We decided to use GitHub Flow[2], a straightforward development workflow.

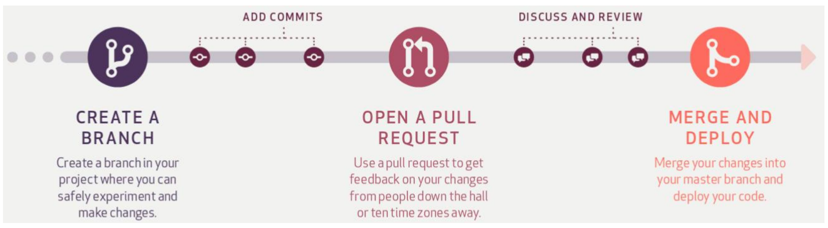

ADD COMMITS
DISCUSS AND REVIEW
CREATE A BRANCH
Create a branch in your project where you can safely experiment and make changes.
OPEN A PULL REQUEST
Use a pull request to get feedback on your changes from people down the hall or ten time zones away.
MERGE AND DEPLOY
Merge your changes into your master branch and deploy your code.

FIGURE 6.1: GitHub Flow illustrated (Source [2])

Since the effective technology will be fixed later in the project, concrete coding guidelines, tools, metrics and an error policy will be defined when appropriate.

6.4 Testing

All functionality must be automatically testable using continuous integration. Any non-trivial function/method must be verified with unit tests.

Integration tests verify extended test scenarios.

A minimal performance analysis will be carried out at the end of the project.

Bibliography

[1] Atlassian Inc. Open Source Services by Atlassian Inc. https://developer.atlassian.com/opensource/, 2017.

[2] Github Inc. Github Flow. https://guides.github.com/introduction/flow/, 2013.

[3] A. Steffen. Task Description "xmpp-grid broker". https://xdg.redbackup.org/documents/task-description.pdf, 2017.

List of Figures

List of Tables

List of Abbreviations / Glossary

A.3 Development Guide

Development Guide

Tools

- Git >= 2.0 for version control

Writing Guidelines

In order to have a consistent style of writing, we defined the following guidelines. These guidelines apply to all documents related to the redbackup project.

- Keep it brief, clear and objective
- Write short and straightforward sentences
- Do not use synonyms for concepts etc. (always use the same wording, e.g. 'client' or 'node')
- Abbreviations must be introduced the first time they occur in the text (except well-known ones)
- Prevent ambiguity in sentences
- Use personal style ("we") whenever appropriate; usually for the description of our work.
- When a gender-specific pronoun is required, use "he/she".
- Use present tense except for the description of (our) completed work.

Definition of Done

To maintain our high quality needs, we determined following definition of done guidelines:

- All functionality conforms to the specification. Any deviations must be discussed and decided by the team.
- A review is performed and accepted in a pull request.
    - The source code is reasonably documented.
    - No code is commented out.
    - No warnings and errors by the compiler or any other quality tool.
    - Reasonable unit and integration tests exist and pass.
    - All documentations are up to date including the project website.
    - The complete continuous integration pipeline works.
    - The code is formatted according to the guidlines (i.e. according to RustFmt)
- The corresponding branch is merged into the stable branch (e.g. master).
- All time is logged.

A.4 Architectural Decisions

Architectural Decisions

Architecture Style

There are three common variants to participate in XMPP communication and manage server configurations: As XMPP-server plugin, XMPP component or an XMPP Client/Bot.

The implementation style fundamentally restricts the set of implementation languages and has a profound impact on the fundamental architecture.

The following aspects must be taken into account to find the most suitable architecture style:

- All required management functionality must be supported over the available APIs and protocols.
- Compatibility with most XMPP servers
- Keep the implementation complexity as low as possible

Considered Options

- XMPP server plugin, e.g. for Openfire Plugin.
- XMPP Component (XEP-0114).
- XMPP Client/Bot

Decision Outcome

Chosen option: XMPP Client/bot, because it is not coupled to a specific XMPP server as the Server Plugin and supports strong authentication, in contrast to the XMPP Component.

Positive Consequences: - A normal XMPP client library can be used to implement the application.

Negative consequences: - The application must run in a daemon, which must handle concurrent XMPP communication.

Pros and Cons of the Options

Server Plugin

- Good, because all features could be implemented directly on the XMPP server.
- Good, because there is minimal protocol overhead and abstraction.
- Bad, because a very high coupling to a specific XMPP server is required and compatibility/interoperability is therefore limited
- Bad, because the high coupling to a specific XMPP server usually limits the possible implementation language.

XMPP Component

- Good, because the application style fits very well in the Components model.
- Bad, because the specification of Components is marked as *Historical* and might therefore not be implemented by many XMPP Servers.
    - Note: Openfire supports Components
- Bad, because some XMPP client libraries might support Components.
- Bad, because the authentication mechanisms might not suffice the required standards of the ietf draft.
- Bad, because it uses a own handshake based digest authentication message.

XMPP Client/Bot

- Good, because a Bot is basically a normal XMPP client, which is supported by every XMPP server
- Good, because all XMPP client libraries implement this feature.
- Good, because secure connections to the XMPP server are supported (SASL).
- Bad, because the application is conceptually not a normal XMPP-Client.

A.5 Time Accounting

TODO

A.6 Meeting Minutes

Meeting 2018-02-19

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Administrative tasks
2. Where to start
3. Date and time for the next meeting
    - 2018-02-26, 09:00 in SFFs office

Discussions / Decisions

1. Administrative tasks
    - All documentation artifacts will be published on the project website.
    - We will use JIRA for project planning.
    - We will decide later, which continous integration tools we use.
        - The decision must allow the INS to take over the project after the BA.
        - reasons for the decision must be documented.
    - Decisions on the programming language and frameworks are made later.
        - Our experience and productivity in a given eco system must be considered as well.
2. Where to start
    - Read the draft XMPP for Security Information Exchange.
    - Learn more about XMPP (Read the specs and the mentioned XEPs).
    - IODEF payloads are not the main focus of this project.
    - It would be nice if a first draft is ready for the IETF Hackathon, starting on March 17.
    - Main goals of the project:
        - Design a solid architecture (Openfire plugin or standalone?).
        - Implement the requirements according to the IETF Draft:
            - Vanilla XMPP with Discovery und Publish/Subscribe XEPs.
            - Define Topics/Nodes and manage permissions.

        - Administrative utilities such as purge, list topics, show number of items, identifier etc.
    - Most is already implemented in the form of a proof of concept.
3. Date and time for the next meeting:
    - 2018-02-26, 09:00 in SFFs office

Upcoming absences

no upcoming absences

Meeting 2018-02-26

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Open questions regarding the task
2. Date for oral exam
3. Is there a presentation during the semester?
4. Date and time for the next meeting

Discussions / Decisions

1. Open questions regarding the task
    - Some requirements can influence the chosen architecture fundamentally (e.g. to write a "bot", a component or a plugin)
        - a "normal" user (in the "bot" or the component variant) might have limited access to some node. According to SFF, this issue can be ignored because, in a real-world application, such behaviour should be limited by strict policies.
        - SFF emphasises that the authentication mechanisms use must be robust and certificate based.
    - Next steps:
        - Write User stories
        - Draw basic architecture in C4-Diagrams
        - Risk analysis (e.g. abuse cases)
        - Evaluation of XMPP servers and libraries. SFF notes that we should not spend too much time evaluating the server and assume that OpenFire fulfils most requirements.
    - Issues to address in the XMPP servers and library evaluation:
        - Can an administrator restrict users to create new topics
        - Recommended features can be checked queried using service discovery. We can also check for undesired configurations (e.g. everyone can publish)
        - Ensure that the libraries support all required functionality, especially authentication!
        - Assess the performance and usability of the libraries

    - It is desirable if the service runs is "always on", e.g. to answer subscription requests. However, this is not strictly required according to SFF.
    - Any kind of Interface is conceivable, a web interface, however, is very flexible. The core functionality does not have to be available separately.
    - The scope of the website is fine according to SFF.
2. License
    - GNU-FDL for the documentation is fine
    - The license for the code will be AGPL but might change depending on the frameworks we use
3. Is there a presentation during the semester?
    - An interim with the internal co-examiner will be carried out.
    - The primary purpose of this presentation is to get familiar with the requirements of the co-examiner (testing, documentation, protocols etc.)
    - Should be carried out if a small demo is ready
4. Date for oral exam
    - If possible, the oral exam will be carried out in early July.
    - We will continuously complete parts of the document to reduce the examination efforts
5. Date and time for the next meeting

Upcoming absences

no upcoming absences

Meeting 2018-03-05

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Discuss and clarify Requirements/Wireframes
2. Current state of the Task Description
3. Date and time for the next meeting

Discussions / Decisions

1. Discuss and clarify Requirements/Wireframes
    - We discussed the open issues in the current requirements draft.
        - The application is meant to run in production and must, therefore, meet strict security requirements
            - Especially TLS >= 1.2 and certificate-based
            - The existing python scripts were the prototype and this project is intended to be a proper implementation
        - Response time is not very crucial for this application as it is designed for maintenance work and not real-time interventions.
        - We must propose an authentication model, e.g. using TLS mutual authentication and exactly one broker user.
        - SFF notes that SASL is quite complex. We might not need it although the IETF draft explicitly requires it.
        - The number of consumers and producers depends strongly on the concrete application. In most cases, however, there will be much more publishers than subscribers. SFF gave the following estimates as reference values.
            - > 1000 Producers
            - 100 Subscribers
            - 1-4 Toplevel Topics
            - > 1000 Subtopics (eg. one for every publisher)
            - > 10000 Persisted Items (note that these items might contain large payloads)

        - We plan to include search, filtering and paging functionality for most listings. We will include these features in the next requirements draft.
        - SFF points out that subtopics are missing in the current draft as well as the wireframes. We will include this in the next requirements draft.
        - SFF notes that a role concept might be needed to simplify the administration. We might be able to use existing XMPP features for this (e.g. XEP-0144).
        - According to SFF, Subscription Requests and Validation are nice to have but have low priority.
            - Instead of validation, unsupported functionality should not be visible
        - For deleting persisted items, SFF suggests a "bulk delete" functionality, which allows administrators to delete all items that match certain criteria.
        - SFF prefers a standalone application over a server plugin to reduce coupling.
        - The implementation language is not of paramount importance to SFF although he prefers Python or Java. A single page app written in JavaScript using a Python backend is also a viable option for him.
            - We should not rely on too many third-party libraries that save us several hours during the project but might require extra maintenance effort in the future (SFF gave a Django paging extension as an example)

2. Current State of the Task Description
    - SFF will complete the task description after he receives the revised user stories and wireframes.
3. Date and time for the next meeting
    - 2018-03-12, 09:00 in SFFs office

Upcoming absences

- No weekly meeting on 2018-03-19 (SFF is absent)

Meeting 2018-03-12

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Discuss Requirements / Wireframes
2. Current State of the Task Description
3. Date and time for the next meeting

Discussions / Decisions

1. Discuss Requirements / Wireframes
2. Current State of the Task Description
3. Date and time for the next meeting
    - 2018-03-23, 09:00 in SFFs office

Upcoming absences

- No weelky meeting on 2018-03-19 (SFF is absent)

A.7 Requirements

The following sections describe the primary requirements in the form of user stories [1]. Figure 2 shows an overview of the primary use stories.

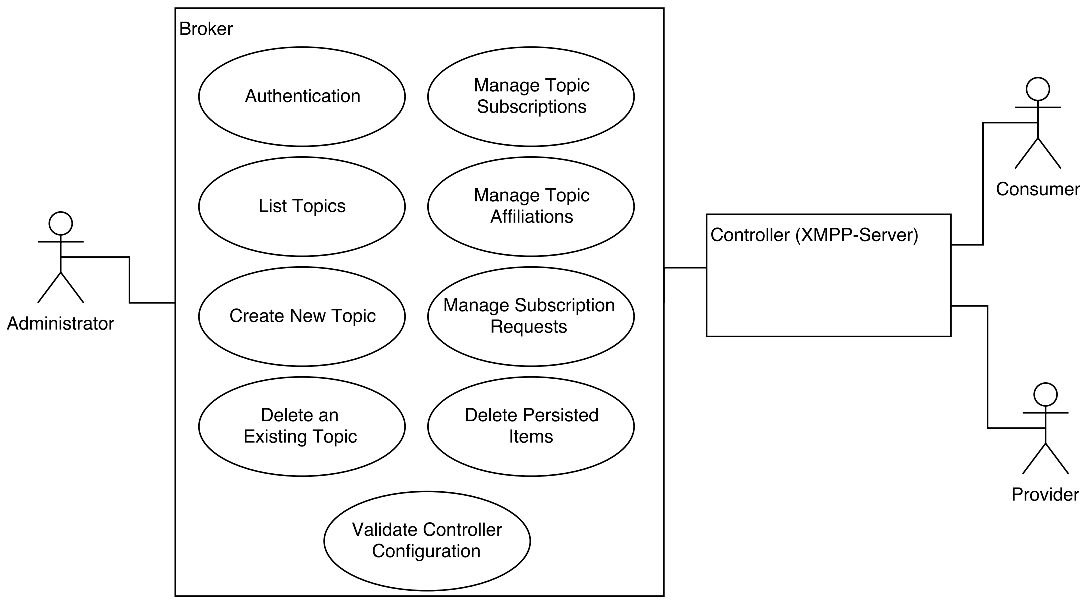


FIGURE 2: UML Use Case Diagram presenting an overview of the primary user stories.

To be clarified

- Non Functional Requirements

A.7.1 Authentication

A.7.1.1 Login

As an Administrator,
I want to log in
so that only I can inspect and manage Topics.

To be clarified

- It would be convenient to not maintain a separate user database by using the same credentials for authentication as for the XMPP server and just forward them. This is from a security/usability perspective not best practice. Are there any specific requirements/wishes?

A.7.1.2 Secure XMPP Connection

As an Administrator concerned with security requirements,
I want to use either SASL EXTERNAL or SASL SCRAM mechanism for authentication -

- preferably the SCRAM-SHA-256-PLUS variant and
- preferably using mutual certificate-based authentication including revocation status checking

- so that the Controller is fully compatible with the XMPP grid draft [4].
To achieve this goal, I am willing to accept:

- More costly and less user friendly authentication
- limited compatibility of supported XMPP servers

A.7.1.3 Logout

As an Administrator,
I want to log out
so that I can terminate a session.

A.7.2 List Topics

A.7.2.1 List All Topics

As an Administrator,
I want to see a list of all Topics of the associated Controller
so that I can quickly assimilate which Topics exist.
To be clarified

- How many topics are realistic? 1-10, 10-100, 100-1000, > 1000? (search required?)

A.7.2.2 List Available Topics With Limited Access

As an Administrator,
I want to see a list of all Topics of the associated Controller to which I have limited access to,
to simplify troubleshooting and locate errors.

To be clarified

- Is this a relevant scenario?

A.7.3 Create a New Topic

As an Administrator,
I want to create a new Topic on the associated Controller
so that I am not tied to a fixed set of Topics.

A.7.3.1 Override Default Topic Configuration

As an Administrator in the process of creating a new Topic,
I want to override the default configuration (e.g. the affiliations)
so that I can restrict access and provide reasonable defaults.

A.7.3.2 Initial Topic Consumers and Providers

As an Administrator in the process of creating a new Topic,
I want to specify an initial set of Consumers and Providers
so that I can restrict access to that Topic and provide reasonable defaults.

A.7.4 Delete an Existing Topic

As an Administrator,
I want to delete an existing Topic on the associated Controller
so that I can get rid of obsolete Topics.

A.7.4.1 Fault Prevention On Delete

As an Administrator in the process of deleting a Topic,
I want a mechanism to prevent me from deleting the wrong Topic on the associated Controller
(e.g. require me to enter the name of the Topic manually).

A.7.5 Manage Topic Subscriptions

A.7.5.1 List Consumers

As an Administrator,
I want to list all Consumers (including their JIDs) of a given Topic on the associated Controller,
so that I can verify that specific Consumers are subscribed, and others are not.
To be clarified

- How many Consumers are realistic? 1-10, 10-100, 100-1000, > 1000? (search required?)

A.7.5.2 Inspect Detailed Subscription Configuration

As an Administrator,
I want to inspect the detailed Topic subscription configuration of a given Consumer, so that I can reproduce and reason about the receipt of data on that Consumer and find potential misconfiguration.

A.7.5.3 Partially Modify Subscription Configuration

As an Administrator,
I want to modify parts of the Topic subscription configuration of a given Consumers, so that I can fix misconfiguration.

A.7.5.4 Unsubscribe Consumer

As an Administrator,
I want to manually unsubscribe a specific Consumer from a particular Topic on the associated Controller,
so that I can remove obsolete or undesired subscriptions.

A.7.5.5 Subscribe Consumer

As an Administrator,
I want to manually subscribe a specific Consumer on a particular Topic on the associated Controller,
so that I can faster setup and manage Consumers.

A.7.6 Manage Topic Affiliations

A.7.6.1 Inspect Affiliations

As an Administrator,
I want to list all Affiliations (JID and "Role") for a particular Topic on the associated Controller
so that I can find potential misconfiguration.
To be clarified

- How many Consumers/Publishers are realistic?

A.7.6.2 Modify Affiliations

As an Administrator,
I want to modify the Affiliation ("Role") of a given JID for a particular Topic on the associated Controller
so that I can fix potential misconfiguration.

A.7.6.3 Fault Prevention When Modifying My Affiliation

As an Administrator in the process of modifying my Affiliation for a particular Topic on the associated Controller,
I want a mechanism to prevent me from accidentally downgrading my rights.

A.7.6.4 Meaningful Error For Topics With Limited Access

As an Administrator,
I want to receive a meaningful error message when inspecting a node to which I have limited access
so that I can quickly comprehend why the configuration options are limited.

A.7.7 Manage Subscription Requests

A.7.7.1 List Subscription Request

As an Administrator,
I want to list pending subscription requests for a given Topic
- given that this feature is supported by the associated Controller -
so that I can quickly assimilate pending requests.

A.7.7.2 Accept Subscription Request

As an Administrator,
I want to accept a pending subscription request for a given Topic
- given that this feature is supported by the associated Controller -
to enable more dynamic access models than just maintaining a black- or whitelist.

A.7.7.3 Reject Subscription Request

As an Administrator,
I want to reject a pending subscription request for a given Topic
- given that this feature is supported by the associated Controller -
so that I can deny user access in accordance with the XMPP standards.

A.7.8 Delete Persisted Items From a Topic

A.7.8.1 Delete a Persisted Item From a Topic

As an Administrator,
I want to delete a particular persisted item from a specific Topic
- given that this feature is supported by the associated Controller -
so that I can clean up test items and remove obsolete or corrupted items.

A.7.8.2 Purge All Persisted Items From a Topic

As an Administrator,
I want to purge persisted items from a specific Topic
- given that this feature is supported by the associated Controller -
so that I can clean up test items and remove obsolete or corrupted items.

A.7.9 Validate Controller Configuration

A.7.9.1 Validate Supported XEPs Configurations

As an Administrator,
I want to validate that a minimum set of XEPs are supported by the associated Controller
so that I can quickly identify incompatibilities.

A.7.9.2 Validate Optional XEP Implementations

As an Administrator,
I want to validate that the required features that are marked as optional or recommended in the XEPs are implemented by the associated Controller
so that I can quickly identify incompatibilities.

A.8 Wireframes

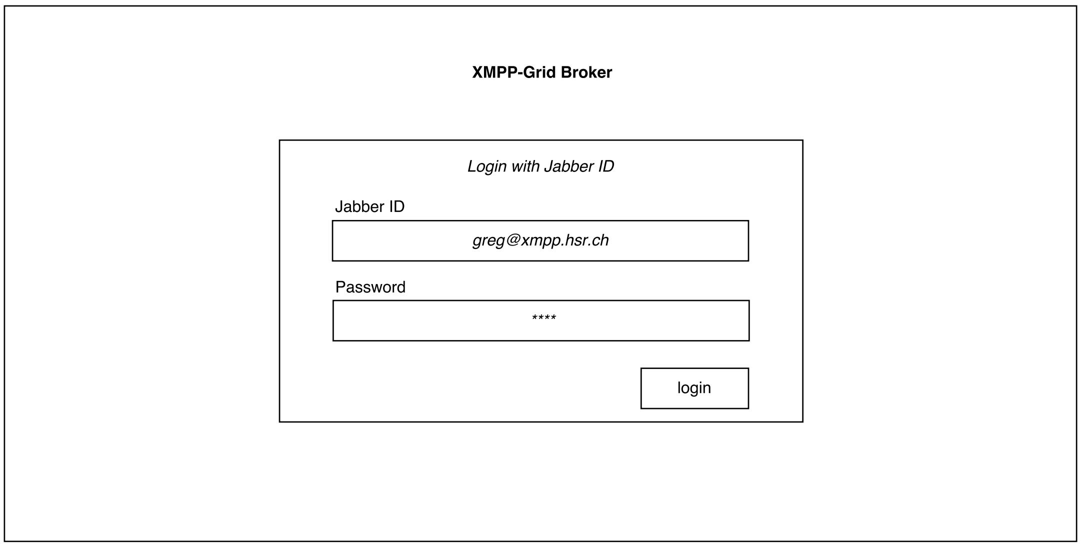


FIGURE 3: Login-Screen Wireframe

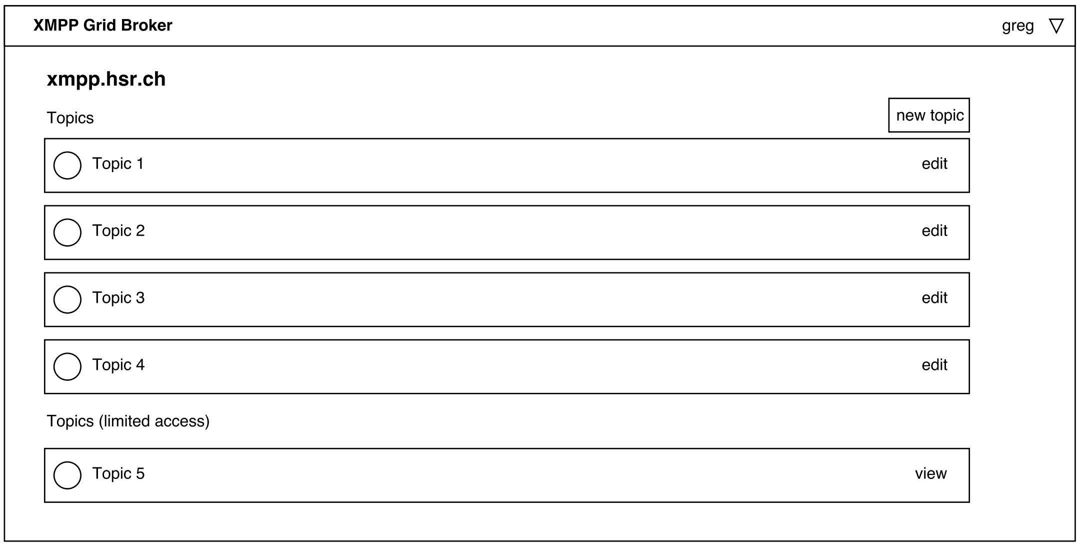


FIGURE 4: Controller Overview Wireframe

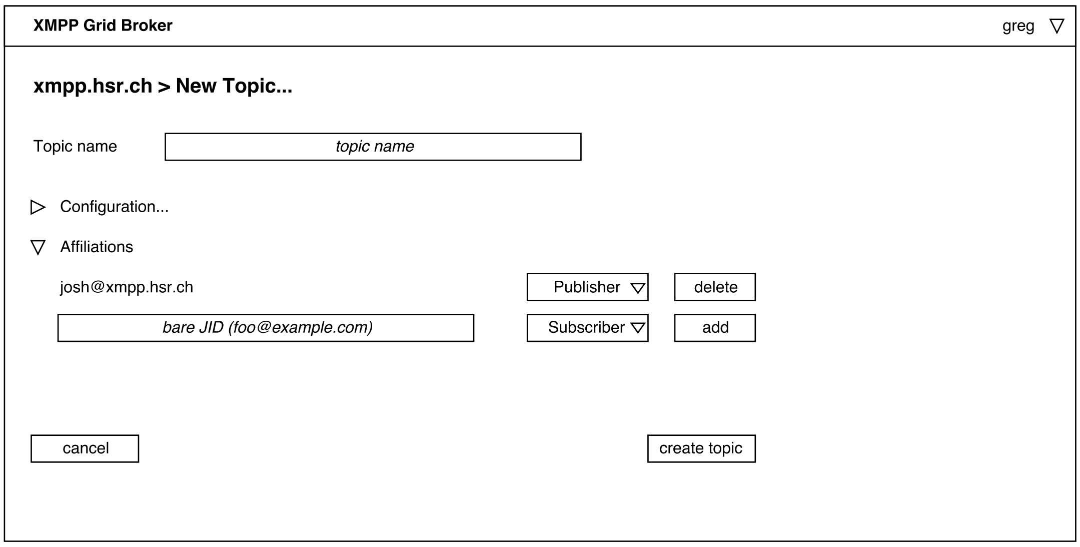


FIGURE 5: New Topic Wireframe

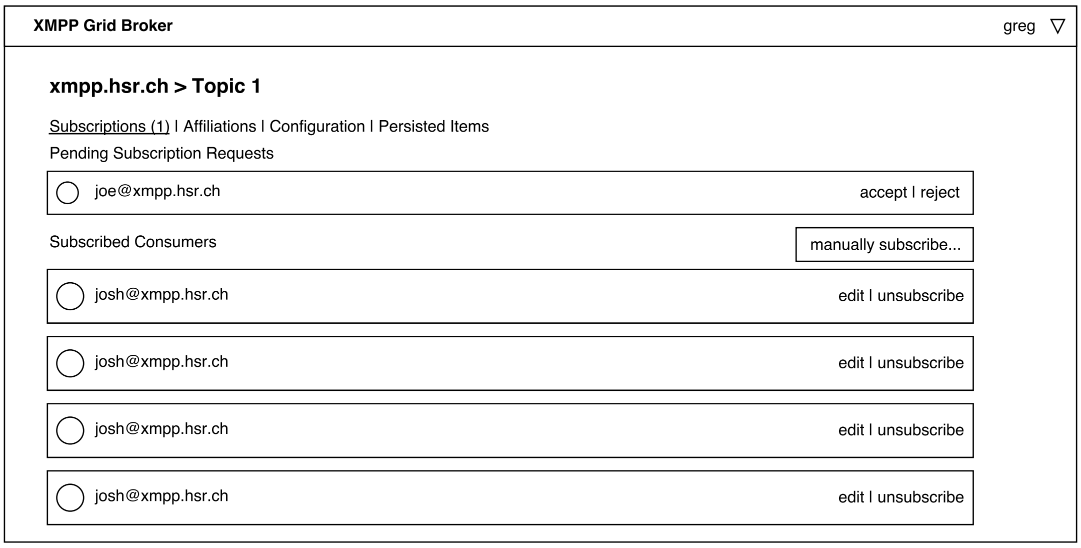


FIGURE 6: Topic Subscriptions Wireframe

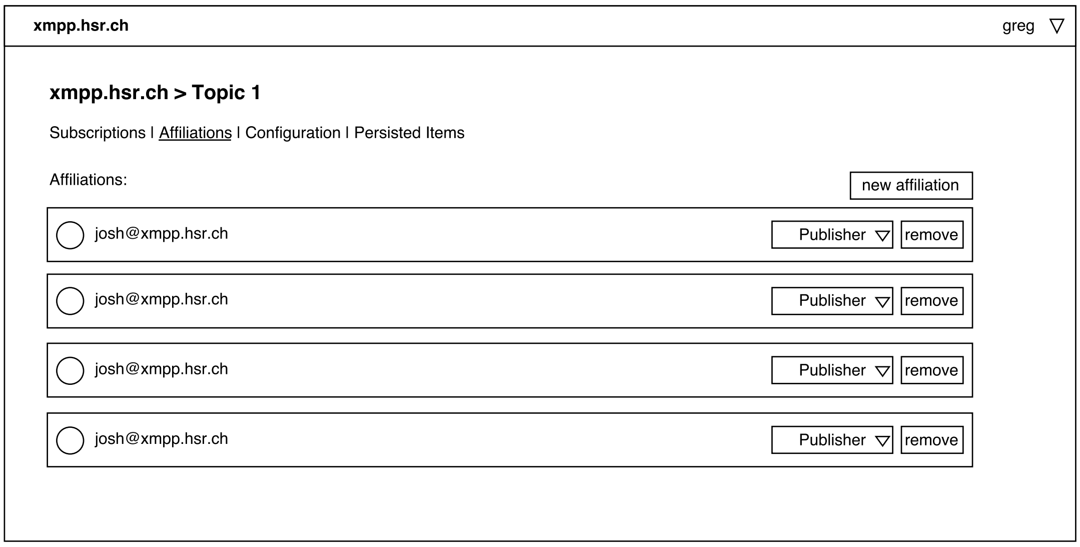


FIGURE 7: Topic Affiliations Wireframe

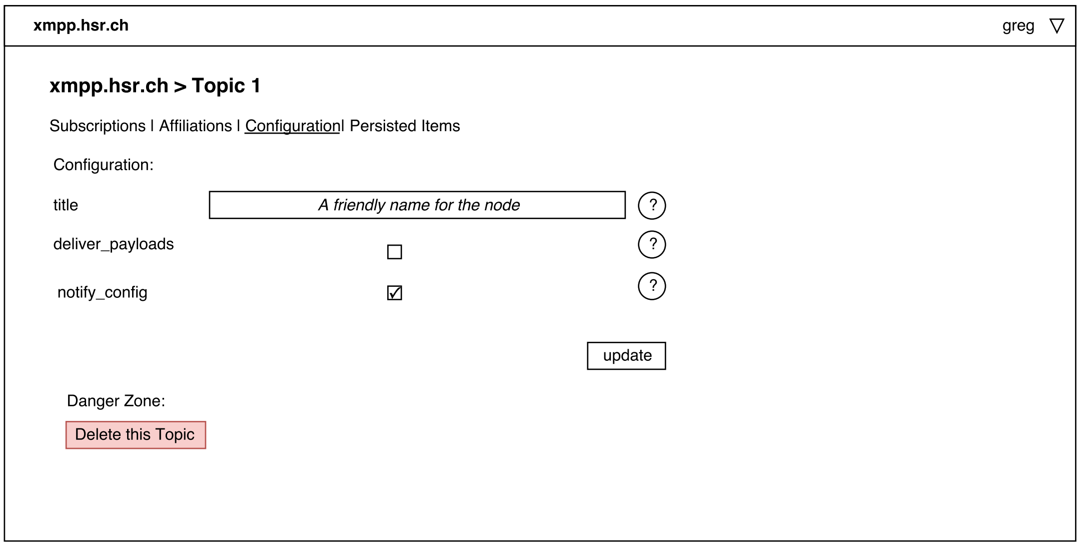
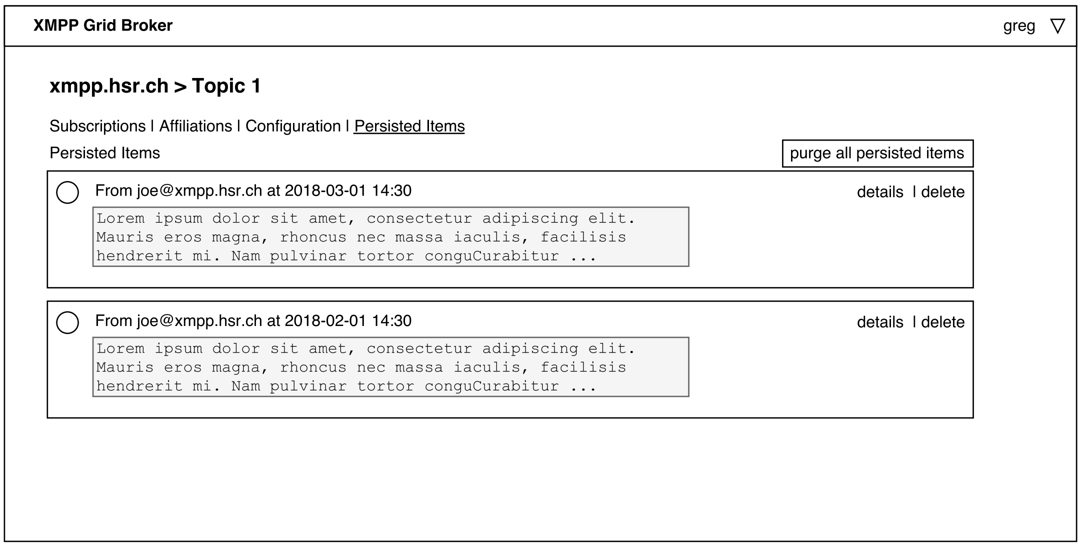


FIGURE 8: Topic Configuration Wireframe


XMPP Grid Broker
greg ▽
xmpp.hsr.ch > Topic 1
Subscriptions | Affiliations | Configuration | Persisted Items
Persisted Items
purge all persisted items
From joe@xmpp.hsr.ch at 2018-03-01 14:30
details | delete
From joe@xmpp.hsr.ch at 2018-02-01 14:30
details | delete

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris eros magna, rhoncus nec massa iaculis, facilisis
hendrerit mi. Nam pulvinar tortor conguCurabitur ...
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Mauris eros magna, rhoncus nec massa iaculis, facilisis
hendrerit mi. Nam pulvinar tortor conguCurabitur ...
```

FIGURE 9: Persisted Items Wireframe

A.9 Comparison of XMPP Server and Libraries

A.9.1 Server

A.9.2 Libraries

TABLE 1: Comparison of XMPP Servers which might be used as Controller

Features / Name	Prog. Language	Plugin Architecture	Req. XEPs	Authentication options
Openfire	Java	Java jar[a]	All fully supported	?
Prosody	Lua	Luascript[b]	XEP-0059 not supported[c]	?
ejabberd	Erlang	Erlang/Elixir[d]	Limited SASL[e]	?

TABLE 2: Comparison of XMPP Client Libraries

Features / Name	Prog. Language	Plugin Architecture	Req. XEPs	Authentication options
SleekXMPP	Python 2	Yes[a]	XEP-0060 client only[b]	?
SliXMPP	Python 3	Yes[c]	XEP-0060 client only[d]	?
aioxmpp	Python 3.4	Yes[e]	Yes[f]	?
Smack	Java	Yes[g]	Yes[h]	?
Babbler	Java	Yes[i]	Yes[j]	?
XMPP-FTW	Javascript (Browser)	Yes[k]	Yes[l]	?

Declaration of Authorship

We, Fabian HAUSER and Raphael ZIMMERMANN, declare that this thesis and the work presented in it are our own, original work. All the sources we consulted and cited are clearly attributed. We have acknowledged all main sources of help.

Fabian Hauser

Raphael Zimmermann

Rapperswil, March 6, 2018