Comprehensive TEX Archive Network Developer's Guide

Version 3.*

Gerd Neugebauer

December 15, 2025

The Comprehensive TEX Archive Network (CTAN) is the major repository for the TEX world. It offers TEX distributions, packages, and more. CTAN provides access via the Web. This site has needed a relaunch. This document describes the consideration on the relaunch of the Web site for CTAN and how to perform certain tasks for installation, operation, and maintenance.

The drawing of the TEX lion shown on the cover page has been provided by Duane Bibby.

Authors

Gerd Neugebauer

gene@ctan.org

Contents

1. Introduction

For long years the Comprehensive Archive Network (CTAN) has served the TEX world as the major repository for all kinds of material. The actual servers have changed over the years.

End of 2011 an activity has been started to relaunch the Web presence of CTAN. The pages used at this time where produced and hosted by Jim Heffron as a private project and not officially by the CTAN team since the time Jim has left the CTAN team. These pages focused on the presentation of the TEX *Catalogue* and not on CTAN as such.

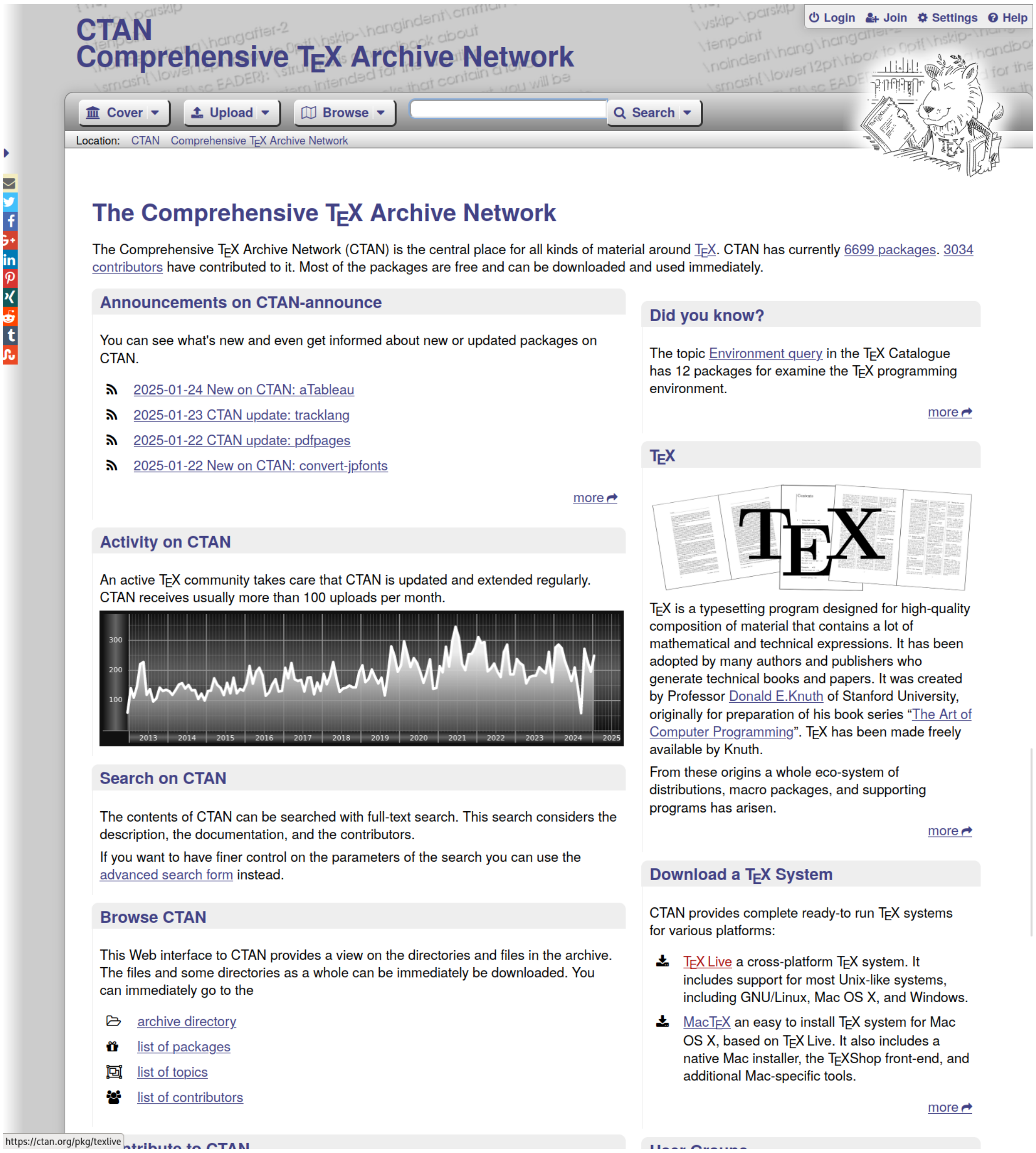
In December 2012 the redesigned Web site had it's public launch. The home page can be seen in figure 1.1.

Now it has served its purpose for several years. The technology has advanced. New frameworks have shown up and the principles for a modern Web application with high usability standards have made some progress. The appearance of release 3 is shown in figure 1.2.

1.1. Goals

The Web site is accompanied with a few goals. Those goals are:

1. Presentation of a few static pages describing CTAN and such.
2. Browsing the CTAN archive by directory.
3. Browsing the CTAN archive by the Catalogue: packages, authors, topics.
4. Searching the CTAN.
5. Providing an *upload*ing form for submitting packages.
6. Providing a *registration* form for CTAN *mirrors*.
7. Allow users to interact with the site by giving feedback, pointing out bugs and corrections, and possibly using additional community features.



Location: CTAN Comprehensive TeX Archive Network

The Comprehensive TeX Archive Network

The Comprehensive TeX Archive Network (CTAN) is the central place for all kinds of material around TeX. CTAN has currently 6699 packages. 3034 contributors have contributed to it. Most of the packages are free and can be downloaded and used immediately.

Announcements on CTAN-announce

You can see what's new and even get informed about new or updated packages on CTAN.

- 2025-01-24 New on CTAN: aTableau
- 2025-01-23 CTAN update: tracklang
- 2025-01-22 CTAN update: pdfpages
- 2025-01-22 New on CTAN: convert-jpfonts

more

Activity on CTAN

An active TeX community takes care that CTAN is updated and extended regularly. CTAN receives usually more than 100 uploads per month.

Search on CTAN

The contents of CTAN can be searched with full-text search. This search considers the description, the documentation, and the contributors.

If you want to have finer control on the parameters of the search you can use the advanced search form instead.

Browse CTAN

This Web interface to CTAN provides a view on the directories and files in the archive. The files and some directories as a whole can be immediately be downloaded. You can immediately go to the

- archive directory
- list of packages
- list of topics
- list of contributors

Did you know?

The topic Environment query in the TeX Catalogue has 12 packages for examine the TeX programming environment.

more

TeX

TeX is a typesetting program designed for high-quality composition of material that contains a lot of mathematical and technical expressions. It has been adopted by many authors and publishers who generate technical books and papers. It was created by Professor Donald E.Knuth of Stanford University, originally for preparation of his book series "The Art of Computer Programming". TeX has been made freely available by Knuth.

From these origins a whole eco-system of distributions, macro packages, and supporting programs has arisen.

more

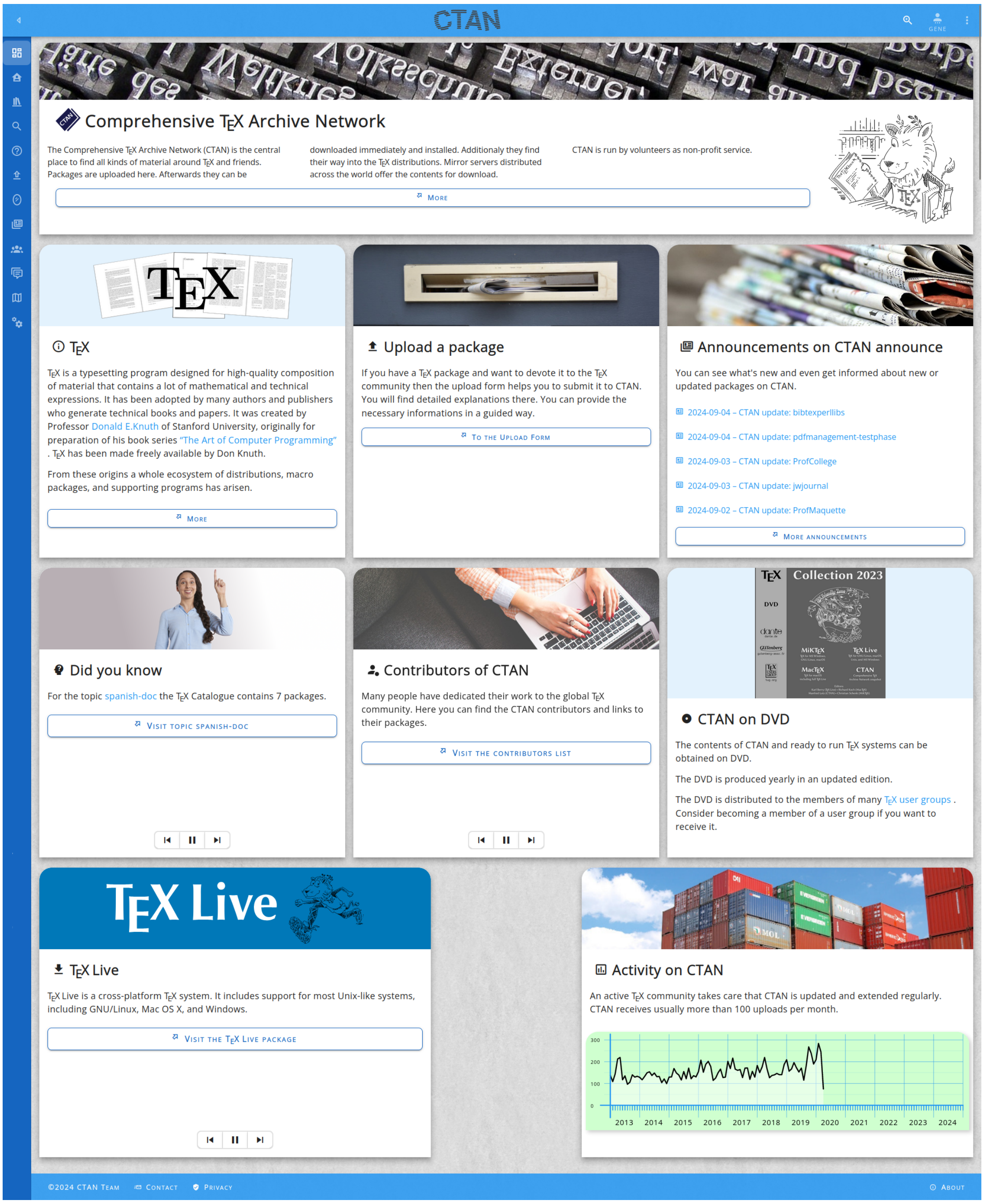
Download a TeX System

CTAN provides complete ready-to run TeX systems for various platforms:

- TeX Live a cross-platform TeX system. It includes support for most Unix-like systems, including GNU/Linux, Mac OS X, and Windows.
- MacTeX an easy to install TeX system for Mac OS X, based on TeX Live. It also includes a native Mac installer, the TeXShop front-end, and additional Mac-specific tools.

more

Figure 1.1.: The CTAN home page in version 2.7



Comprehensive TeX Archive Network

The Comprehensive TeX Archive Network (CTAN) is the central place to find all kinds of material around TeX and friends. Packages are uploaded here. Afterwards they can be downloaded immediately and installed. Additionaly they find their way into the TeX distributions. Mirror servers distributed across the world offer the contents for download.

CTAN is run by volunteers as non-profit service.

TeX

TeX is a typesetting program designed for high-quality composition of material that contains a lot of mathematical and technical expressions. It has been adopted by many authors and publishers who generate technical books and papers. It was created by Professor Donald E.Knuth of Stanford University, originally for preparation of his book series "The Art of Computer Programming" . TeX has been made freely available by Don Knuth.

From these origins a whole ecosystem of distributions, macro packages, and supporting programs has arisen.

Upload a package

If you have a TeX package and want to devote it to the TeX community then the upload form helps you to submit it to CTAN. You will find detailed explanations there. You can provide the necessary informations in a guided way.

Announcements on CTAN announce

You can see what's new and even get informed about new or updated packages on CTAN.

- 2024-09-04 – CTAN update: bibtexperllibs
- 2024-09-04 – CTAN update: pdfmanagement-testphase
- 2024-09-03 – CTAN update: ProfCollege
- 2024-09-03 – CTAN update: jwjournal
- 2024-09-02 – CTAN update: ProfMaquette

Did you know

For the topic spanish-doc the TeX Catalogue contains 7 packages.

Contributors of CTAN

Many people have dedicated their work to the global TeX community. Here you can find the CTAN contributors and links to their packages.

CTAN on DVD

The contents of CTAN and ready to run TeX systems can be obtained on DVD.

The DVD is produced yearly in an updated edition.

The DVD is distributed to the members of many TeX user groups . Consider becoming a member of a user group if you want to receive it.

TeX Live

TeX Live is a cross-platform TeX system. It includes support for most Unix-like systems, including GNU/Linux, Mac OS X, and Windows.

Activity on CTAN

An active TeX community takes care that CTAN is updated and extended regularly. CTAN receives usually more than 100 uploads per month.

Figure 1.2.: The CTAN home page

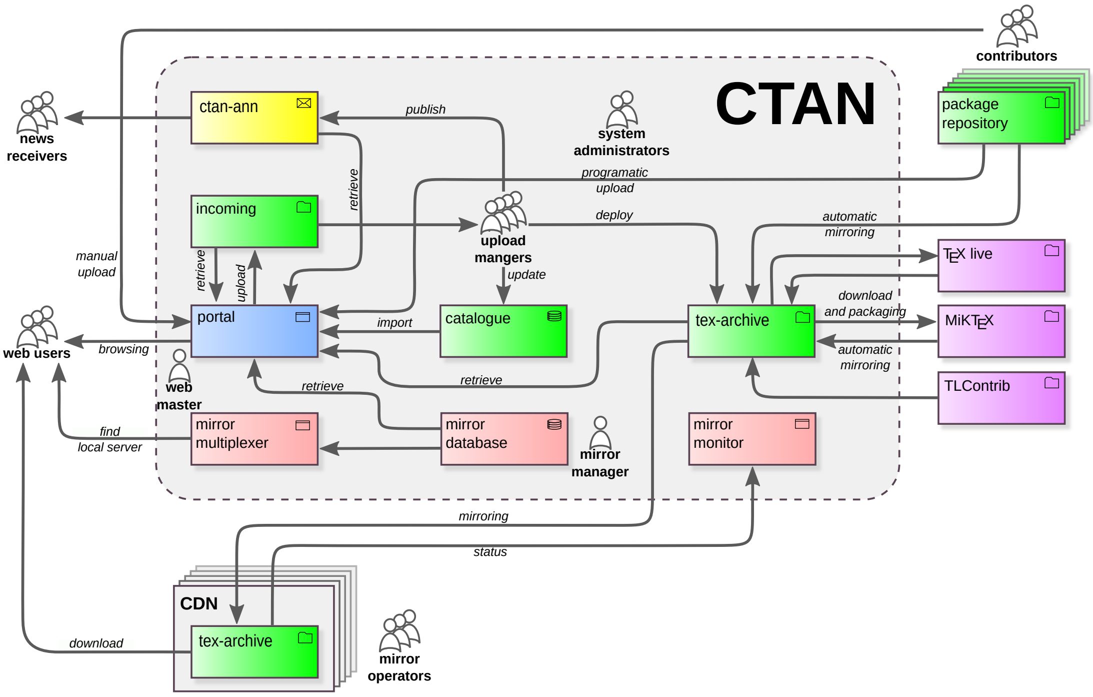


Figure 1.3.: Interaction flows overview

1.2. Improvements

The relaunch might be accompanied by some community functions which goes into the direction of a communication platform and less into mere presentation of some information.

1.3. Interaction flows

missing

2. Architecture

In this chapter we describe several aspects of the architecture of the CTAN site.

2.1. Users and Roles

Various roles are relevant in the context of the CTAN site. An overview is shown in figure 2.1. The arrows denote the inheritance relation as known from UML [RJB04].

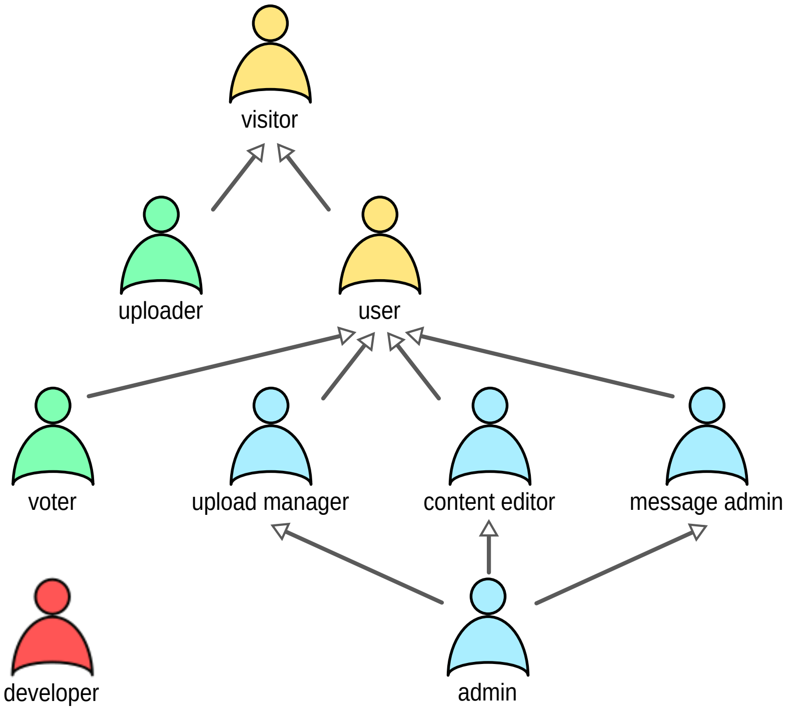


Figure 2.1.: User roles of the CTAN site

2.1.1. Visitor

Visitor is one of the roles in the context of CTAN. A visitor uses the CTAN site. This can be done anonymously. Then the visitor has not further special rights.

2.1.2. User

User is one of the roles in the context of CTAN. A user is authenticated. Thus the additional rights and functions of an authenticated user are available. A user is a visitor with additional permissions.

A user has to go through a registration process and a login.

2.1.3. Uploader

An uploader is a visitor who submits a package to CTAN. This can be performed anonymously or authenticated. When the uploader is authenticated then some more functionality is available.

2.1.4. Admin

Admin usually perform their tasks in the background behind the CTAN site. Nevertheless some functionality is also provided via the Web site. Thus the Admin is a user with additional rights.

2.1.5. Upload Manager

The upload manager takes care of upload. When a contributor has uploaded a new package or an update of a package. The upload manager receives am email notification about an upload. Then the uploaded package can be retrieved from the incoming directory and move the package files to the TEX-archive directory and the catalogue repository.

2.1.6. Mirrors Manager

The submission of new mirrors can be performed with the CTAN site. The mirror manager updates the active mirrors in the mirrors database.

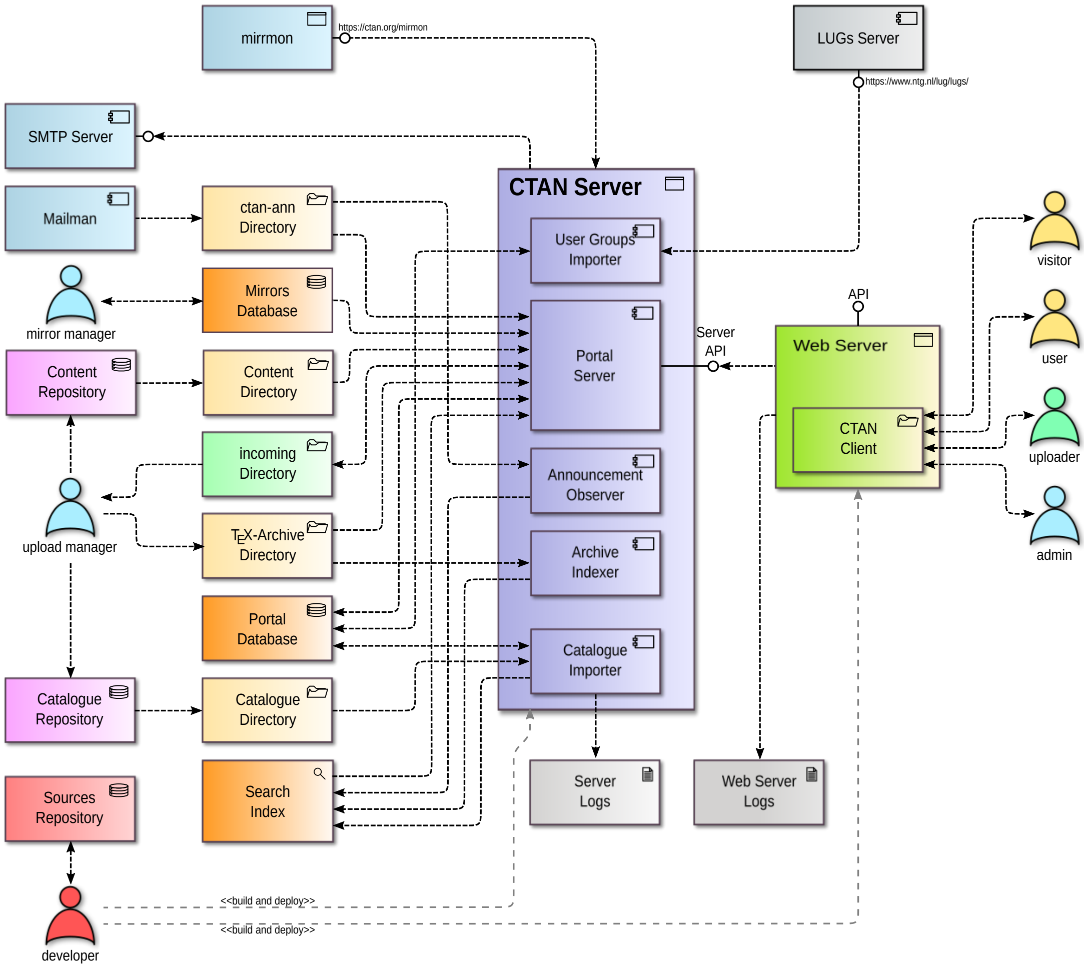


Figure 2.2.: Data flows in the context of the CTAN site

2.1.7. Developer

The CTAN site consists of software. For the completeness we have the role of a developer.

2.2. Data flows

Data from various sources play a role in the context of the CTAN site. An overview is shown in figure 2.2. The components and data flows are described in the following section.

2.2.1. TEX-archive directory

The TEX-archive directory (`/home/ftp/pub/tex`) is a directory which contains the raw files of CTAN. The TEX-archive directory is manually filled with the contributions by the upload managers. In addition some external sources are mirrored into this directory structure.

The TEX-archive directory is traversed by the Archive Indexer. The search index is updated accordingly.

The upload managers place the contributions in the TEX-archive directory. From there the files are accessible through the CTAN site or the mirror servers.

2.2.2. Site database

The site database contains the data for the CTAN site. To allow an efficient access the CTAN catalogue is copied into the site database (database: `portalctan; host: localhost; port: 5432`).

The data has the master in the XML files of the catalogue. It is located on an instance of PostgreSQL.

Other data is primarily contained in the site database. This is especially the attributes of the users and other data not originated in the catalogue XML files.

2.2.3. Incoming directory

The incoming directory (`/serv/www/www.ctan.org/incoming`) is used by the CTAN site to store incoming submissions. The upload managers prepare the submission and finally place them in the TEX archive directory and update the Catalogue in the Catalogue Repository.

2.2.4. Content Repository

The content repository contains the pages and fragments in the supported languages. This allows an update without rebooting the web application.

The content repository is checked out into the content directory regularly.

The content repository is a Git repository. It is located at

`https://gitlab.com/comprehensive-tex-archive-network/ctan-content`.

2.2.5. Content directory

The CTAN site is designed to be multi-lingual. Currently the languages English and German are present. The pages and fragments for the supported languages are stored in the content repository and checked out into a working directory – the content directory (`/serv/www/www.ctan.org/ctan-content`). From there the CTAN site read the texts and provides them for the Browser UI.

The content directory is updated on a regular schedule. This means a checkout is scheduled.

2.2.6. Catalogue repository

The catalogue is kept in the catalogue repository. The catalogue repository is am Apache Subversion repository. The upload managers maintain the package meta-data in the catalogue. The catalogue is checked out regularly into a workspace in the catalogue directory.

2.2.7. Catalogue directory

The catalogue directory contains a workspace copy of the catalogue repository. It is located in `/serv/www/www.ctan.org/texcatalogue/entries`. From here the site database and the search index are updated.

2.2.8. Search index

The CTAN site uses Apache Lucene as search engine. This search engine maintains a search index in the directory `/serv/www/www.ctan.org/index`.

2.2.9. Server logs

The back-end server is an instance of Apache Tomcat. It writes log files.

2.2.10. Web-server logs

The Web server acts primarily as proxy server. The Web server is an instance of the Apache HTTPD. It writes log files.

In addition the log for the Catalogue update are produced here.

2.2.11. Mail server

The mail server is an SMTP server to deliver outgoing mails from the CTAN site.

2.2.12. Mailman

Mailman 3 is used to manage mailing lists. This holds especially for the mailing list `ctan-ann@ctan.org` which contains announcements for new or updated package uploads on CTAN.

Regularly the mails sent out are stored in the directory `ctan-ann`.

2.2.13. ctan-ann directory

The ctan-ann directory (`/serv/www/www.ctan.org/ctan-ann`) contains copies of the announcements of package uploads on the mailing list ctan-ann.

2.2.14. Mirrors database

The mirrors database (database: `ctan; host: localhost; port: 5432`) contains the mirrors servers. It is located on an instance of PostgreSQL.

2.2.15. Lugs server

The local TEX user groups (LUGs) are maintained by the NTG. The CTAN site retrieves this data from `https://www.ntg.nl/lug/lugs/` and offers a page with it.

2.3. Architecture Decision Records

Each software system has an architecture – may it be carefully planned or just grown "by accident". We want to document the major architecture decisions. The theoretical background is given as "architecture decision records". Several templates have been published for ADRs. Our ADRs are based on the work of Jeff Tyree and Art Akerman (cf. [TA05]). The template is mainly taken from

`https://github.com/joelparkerhenderson/architecture-decision-record`

which is described in [TA05].

We have a template in the sources under

```
doc/ctan-developers-guide/architecture/adr/_template.tex
```

This template contains the sections and explanations.

2.3.1. Back-end programming language: Java

Issue

The back-end server has to be written in a programming language. This programming language has to be selected.

Decision

The programming language Java is used for the back-end server.

Status

Decided

Group

Back-End

Constraints

1. The programming language for the back-end should be available as Open-Source.
2. The programming language for the back-end should be widely used.

Positions

1. Java is an well-established programming language. It is often used to implement micro services and services.
2. Groovy is a programming language running on the JVM. It is similar to Java and tries to cope with some of the problems in Java.
3. Kotlin is a modern programming language in the Java world.

4. JavaScript is a programming language which started as client-side language. Nowadays it is used on the server-side aswell.
5. TypeScript is a programming language which has added better support for object-orientation to JavaScript.
6. PHP is a programming language which started as tool to build Web pages. A huge number of libraries is available for PHP.
7. Python is a programming language wich supports object-oriented programming, aspect-oriented programming and functional programming. If tries to promote a readable style of source code.
8. C++ is a object-oriented development based on the programming language C.

And many more programming languages come to mind. But all of them are less appropriate.

Argument

- Java is old enough such that the deficiencies are well-known.
- Java has Open-Source implementations.
- Java is known by the main developer.

2.3.2. Back-end framework: Dropwizard

Issue

The overall architecture of a Web application like the CTAN site consists of back-end server and a client part. The back-end server should be based on an appropriate framework to ease the task of programming.

Decision

The back-end framework Dropwizard is used.

Status

Decided

2.3.3. Group

Back-End

Constraints

1. The framework should be Open-Source.
2. The programming language for the back-end server is Java – or at least it runs in a JVM.

Positions

1. Dropwizard
2. Spring Boot
3. Grails

Argument

- Dropwizard has good support as one of the major frameworks.
- Dropwizard has less features preconfigured. Thus you can freely decide what to add.
- Spring Boot has more features enabed by default. Thus the startup appears to be slower.
- Grails is based on Spring Boot. It is tailored towards server-side rendering.

2.3.4. Back-end programming library: Lombok

Issue

The back-end server is written in the programming language Java. The programming should be made as simple as possible. Thus a libraries can be considered for this purpose.

Decision

The programming library Lombok is used for the back-end server wherever possible to compensate the deficiencies of the Java programming language.

Status

Decided

Group

Back-End

Constraints

1. The programming library for the back-end should be available under a Open-Source license.
2. The programming library for the back-end should be widely used, i.e. especially it should be independent from the IDE.

Positions

1. Lombok is an well-established programming library. It started as a tool to get rid of getters and setters.
2. Avoid any complications of an additional library and use the features of the Java language purely.

Argument

- Lombok is well established.
- Lombok can be used with a lot of IDEs – like Eclipse and IntelliJ IDEA.
- Lombok is Open-Source.
- Lombok reduces the boilerplate code in Java programs.

2.3.5. Database: Postgres

Issue

The various data items need to be accessed efficiently. This includes the catalogue as well as other entities like users and parameters.

Decision

The data is primarily stored in a SQL database with PostgreSQL as DBMS.

Status

Decided

Constraints

1. The system is self-hosted.
2. The underlying operating system is Linux – most probably Debian.
3. The database should be Open-Source.

Positions

1. Keep the data in files. Possible formats include XML, JSON, or YAML.
2. Use a SQL database. Possible DBMS' are PostgreSQL or MariaDB.
3. Use a non-SQL database.

Argument

- The use of an SQL database allows efficient and fast access to the data. This supports the usability and acceptance by the end user.
- The DBMS should be an Open-Source product to avoid any costs for development and production.
- PostgreSQL DBMS was already in use for the mirror database.
- PostgreSQL was well-known to the persons involved.

Implications

- The workflow of the upload managers is still based on the XML files of the Catalogue. Thus a regular update of the database is required.

2.3.6. Search Engine: Apache Lucene

Issue

Searching is a major feature of the CTAN site. Thus a powerful and fast search engine should be used.

Decision

The search engine Apache Lucene is used.

Status

Decided

Constraints

1. The search engine should be Open-Source.
2. The programming language for the back-end server is Java. The search engine should easily integrate.
3. The search engine should be integrated and run under the control of the CTAN team.

Positions

1. Apache Lucene is a Java library to perform powerful searches. It is integrated into an application.
2. Elasticsearch is a search engine based on Lucene.
3. Apache Solr is a search engine based on Lucene.
4. OpenSearch is a search engine based on Lucene.
5. External search engines like Google or DuckDuckGo can be integrated.
6. Database search can make us of the database indexing features.

Argument

- Lucene is sufficient. The additional features of Elasticsearch, Solr, or OpenSearch are not required.
- Lucene does not need additional processes to be run beside the main server.
- External search engines are not under the control of the CTAN team. A counter example is Bing which has closed down its search API in August 2025.
- Database indexing would require many additional programming effords to have the powerful features present in Lucene.

2.3.7. Front-end: Client-side-rendering

Issue

The generation of the HTML pages can be performed either on the server-side or on the client side. This architecture decision has to be made.

Decision

The rendering is performed in the client. The server has very few knowledge about HTML.

Status

Decided

Group

Front-End

Constraints

1. The resulting site should be optimised for the user. The programmer is secondary.

Positions

1. The rendering is performed mainly on the server (SSR). Complete HTML pages are shipped out.
2. The rendering is mainly performed on the client (CSR). The client communicates with the server via JSON APIs.

Argument

- SSR has the advantage that everything is on the server and can be accessed from the code there.
- SSR has the advantage that the complete pages can be cached. Successive access to the same pages gain performance.
- SSR has the advantage that no active code is executed in the browser. This helps to avoid hacker attacks.
- SSR has the advantage that the site works even if JavaScript is disabled. But JavaScript has become important that many sites massively rely on it.
- CSR has the advantage to separate the concerns. The front-end specific concerns are located in the client code. The back-end specific concerns are located in the server code.
- CSR has the advantage that the site in the browser is more responsive. No long lasting calls to the back-end for the next page are needed. The site appears more responsive.
- CSR has the advantage that modern frameworks already support it.
- CSR has the advantage hat part of the computation is performed in the browser which reduces the load for the server.

2.3.8. Front-end programming language: JavaScript

Issue

The front-end code has to be written in a programming language. This programming language has to be choosen.

Decision

The programming language JavaScript in the version ECMAScript 6 (ES6) is used for the back-end server.

Status

Decided

Group

Front-End

Constraints

1. The programming language for the front-end should be available as Open-Source.
2. The programming language for the front-end must run in many browsers.
3. The programming language for the front-end should be widely used.

Positions

1. JavaScript is an well-established programming language for the Web. The version ES6 is sufficiently modern and out-of-the-box available in many browsers. It is one of the supported languages vor Vue.
2. TypeScript is a modern programming language based on JavaScript. It adds object-orientation to the language. It is one of the supported languages vor Vue.

Argument

- JavaScript in version ES6 is sufficiently modern and immediately available in many browsers. Such that the incompatibilities of former versions are no longer a problem.

2.3.9. Front-end framework: Vuetify

Issue

The overall architecture of a Web application like the CTAN site consists of back-end server and a client part. The front-end client should be based on an appropriate framework to ease the task of programming.

Decision

The front-end framework Vuetify is used.

Status

Decided

Group

Front-End

Constraints

1. The framework should be Open-Source.
2. The programming language for the front-end server is JavaScript.

Positions

1. Vuetify is a framework based on Nuxt and thus Vue. It provides components to be compatible to Google's Material Design.
2. Angular is one of the older frameworks. It has a large user base and many components. Nevertheless it suffers from some ancient design decisions. Especially the separation of structure, design, and functionality can be distracting.
3. React is one of the major frameworks. It is gaining momentum and is a real alternative.
4. Svelte is a newcomer. It has the advantage to be fast. Nevertheless as a newcomer the stability and richness of functions was not given at the time when this decision had to be made.

5. JQuery is a well established framework. It provides a small set of low-level functions to make the same code running under all supported browser versions. It does not provide a design philosophy.

Argument

- Vuetify provides useful components.
- The components are prepared to buils a modern looking UI.
- The bundling of a component in one file helps to keep things organised and to have an easier overview.

2.3.10. Build tool: Gradle

Issue

The build of the system should be automated. For this purpose a build tool should be used. Here the tasks and dependencies are specified. Then the actions for the building can be minimised.

Decision

The build tool Gradle is used.

Status

Decided

Constraints

1. The build tool should be Open-Source.
2. The programming language for the back-end server is Java. The build tool should be aware of Java and support it out of the box.
3. The build tool should be usable outside af the IDE. This means that a command line interface is a must.

Positions

1. Gradle is a modern build tool. The configuration can be written in Groovy or Kotlin.
2. Apache Maven is one of the older tools. The configuration is wrotten in XML.
3. Apache Ant is one of the older tools. It is initially not prepared to cache access to libraries from the net. This requires an additional component like Apache Ivy.
4. GNUmake is a well established build tool. It can be used by a broad range of programming tasks. Nevertheless the use for a Java system requires manual configuration.

Argument

- Gradle is usually faster than Maven.
- Gradle and Maven make use of Maven reporitories for many libraries.
- Ant uses a cache only with the help of Ivy.
- GNUmake does not use the various libraries from the net.

2.3.11. Documentation: LaTeX

Issue

A software project usually has a lot of things which are best preserved in written form. Thus we have to decide which documentation framework to use.

Usually several kinds of documentation have to be considered. For instance

- Architecture documentation
- Source code documentation
- Tickets
- READMEs

Here we only consider stand-alone documents – mainly for architecture documentation. They are larger documents. They have to be readable to understand the larger context. In the end they are provided as PDF file. These PDF files have to be created somehow.

Decision

The documentation framework LaTeX [Lam86] is used for stand-alone architecture documents.

Status

Decided

2.3.12. Group

Documentation

Constraints

1. The documentation framework should be Open-Source.

Positions

1. LaTeX
2. plainTeX
3. HTML
4. Markdown

Argument

- CTAN is a building block in the TeX word. Thus a framework from the TeX eco-system are highly preferred.
- LaTeX is widely used and support for many aspects can be acquired – especially from CTAN.

3. Rules

This chapter collects some of the rules to which the developers should obey.

1. **All rules can be overruled**
   All rules can be overruled. This requires a sufficient justification in written form.
2. **Development language is English**
   The language used throughout this application is English (in the British dialect).

   This applies to the documents describing the application as well the documentation of the sources, the description of the internal aspects, and the naming of files, classes, methods, functions, etc.

   An exception are documents for external audiences. Since the site is multilingual this means that the texts presented in th UI can also be considered to be an exception.
3. **Sematic versioning**
   Whenever versioning is required then we use the scheme of semantic versioning [SEM].
4. **Documentation**
   Documentation is required. For details see section 3.2.

3.1. Definition of done

The term "definition of done" originated in the Scrum methodology [SCR] for software development.

According to the Scrum Guide

> The Definition of Done is a formal description of the state of the
> Increment when it meets the quality measures required for the product.

According to this definition we want to use the following criteria:

1. Unit tests are written and passing.
   - The unit tests for Java should cover more than 70% of the code.
   - The unit tests for JavaScript are optional.

  - The active unit tests all pass.

2. The build passes.
3. The user interface has been tested.
4. The documentation is up-to-date.
5. The generation of Javadoc passed.

3.2. Documentation

Documentation is not optional. Several goals can and should be achieved with appropriate documentation:

Preserving knowledge

New developers need to get used to the standards in an application. Asking the old fellows might not be possible. Oral tradition is time consuming and error prone.

The pure source code shows the how, but not the why. Thus documentation has to fill the gap.

Overall consistency

If rules are formalised and documented then it is easier to achieve overall consistency.

Written documentation helps to think about the rules and discuss them in a team.

Ensure quality

One principle in quality assurance is to do things several times. But sheding light on the same spot from different directions helps to avoid dark spots.

Thus the following activities support this idea:

- Write a specification of what should be achieved.
- Write source code to do what is required.
- Write test cases to check that the goals have been reached. (and test regularily)

3.2.1. General introduction

README.md

missing

3.2.2. Documenting Java code

License header

```
/*
 * Copyright © 2022-2025 The CTAN Team and individual authors
 *
 * This file is distributed under the 3-clause BSD license.
 * See file LICENSE for details.
 */
```

JavaDoc

missing

3.2.3. Documenting Vue code

License header

```
<!--
  -- Copyright © 2022-2025 The CTAN Team and individual items
  --
  -- This file is distributed under the 3-clause BSD license.
  -- See file LICENSE for details.
  -->
```

JSDoc

missing

3.2.4. Documenting JavaScript

License header

```
/*
** Copyright © 2024-2025 The CTAN Team and individual authors
**
** This file is distributed under the 3-clause BSD license.
** See file LICENSE for details.
*/
```

JSDoc

missing

3.2.5. Document the licensing

Especially in an Open-Source context the licensing of the different parts should be made clear.

This application is distributed under the 3-clause BSD license.

The licensing information has to be put in several places. If parts of the software are reused somewhere else then the lisense should not be lost.

- Provide a file `LICENSE` in the top-level directory of the project containing the licensing conditions.
- Each source file containing program code should have a head which names the license under which is distributed.

missing something else?

3.2.6. Documenting LaTeX

License header

```
%% *****************************************************************************
%%
%% Copyright © 2024-2025 The CTAN Team and individual authors
%%
%% This file is distributed under the 3-clause BSD license.
%% See file LICENSE for details.
%%
%% Author: <a href="mailto:gene@ctan.org">Gerd Neugebauer</a>
%%-----------------------------------------------------------------------------
```

missing

3.3. Testing Java code

The sercer-side code is written in Java. This must be tested.

3.3.1. Organising tests

Tests are located in the directories `src/test/java` and `src/test/resources`. The first directory contains the Java code for running tests. The second directory contains additional files needed for the tests.

The directory structure is identical to the package structure of the productive code. The test suite for a class is in the same directory as the class under test – in the test directory. Thus the test can access the methods with appropriate visibility.

The name of the test suite is the name of the class under test with `Test` appended.

3.3.2. Test frameworks

The following frameworks are used for writing tests:

JUnit 5

AssertJ

Mockito

Hamcrest

missing

3.3.3. Given-when-then

The test cases are structured following the given-when-then pattern. The following three sections are used for this purpose:

Given specifies the context of the test case. This section is optional. It can contain one or more conditions combined with **and**.

When specifies the action to be performed for the test. It can be omitted in rare cases. If a sequence of actions is required for the test then the actions can be combined with **and**.

Then specifies the outcome. It may contain conditions describing the direct return value or conditions describe the state after the actions. This section is not optional. It can contain one or more conditions combined with **and**.

The given-when-then pattern should be used in the program code of the test case for structuring. In addition the Javadoc of the test case should should contain the full text form. This is shown in the following example:

```java
/**
 * When the method toMap is called<br>
 * Then the map returns the id and the stopword as keys.
 */
@Test
void testToMap() {

    // Given
    var instance = UserStopword.builder()
        .id(123L)
        .stopword("word")
        .build();
    // When
    var result = instance.toMap();
    // Then
    assertThat(result)
        .hasSize(2)
        .hasFieldOrPropertyWithValue("id", 123L)
        .hasFieldOrPropertyWithValue("stopword", "word");
}
```

missing more

4. Internationalisation

CTAN is prepared to support several languages. Here we describe what needs to be done to add another language.

In this chapter we use the language nl (i.e. new language) in the sample code.

4.1. Defining a language in the server configuration

In the file `src/main/resources/ctan.yml` the supported languages are configured. Here the new language (nl) is added:

```
ctan:
  languages:
    - en
    - de
    - nl
```

The end-point `/api/3.0/site/config` will transport this information to the client.

4.2. Language in the client

4.2.1. Vuetify configuration

The file `src/client/plugins/vuetify.js` contains – among other things – the configuration for the localisation. Here the new language has to be added:

```
  const i18n = createI18n({
    fallbackLocale: 'en',
    legacy: false,
    locale: 'en',
    messages: {
      en,
      de,
      nl
    },
    warnHtmlMessage: false,
  })
```

4.2.2. I18n properties

In the directory `src/client/i18n` there are JavaScript properties defining the language-specific strings.

Edit each of the files and add the language to the key `/LANG` if it is not there already. The value is the name of the language in this language. This means it is not translated automatically!

Then copy one file as starting point for the new language:

```
cp en.js nl.js
```

Then you change the values in this new file. Note

- If the value is enclosed in single quotes (`'`) the the value may span one line only.
- If the value is enclosed in back quotes (`` ` ``) the the value may span several lines.
- Logos can be typeset with entities. See table 4.1.

4.2.3. Components with static texts

Some components have texts hardwired. This has mainly performance reasons. Those components have to be extended with the new language;

DcdCard

TexCard

WantedCard

Page about

4.3. Internationalisation of the help pages

The help pages are located in a Git repository of its own. This repository can be found under

```
git@gitlab.com:comprehensive-tex-archive-network/ctan-content.git
```

The directory `page` contains the help pages. The directory structure corresponds to the URL on the site. The filename has an extension which corresponds to the language. Thus copy each and every file with the extension `.nl` and translate the contents.

entity	logo
`&AMS;`	AMS
`&AMSTeX;`	AmSTEX
`&AmSTeX;`	AmSTEX
`&AMSLaTeX;`	AmSLATEX
`&AmSLaTeX;`	AmSLATEX
`&BibTeX;`	BIBTEX
`&BibLaTeX;`	BIBLATEX
`&ConTeXt;`	ConTEXt
`&eTeX;`	ε-TEX
`&LaTeX;`	LATEX
`&LaTeX2e;`	LATEX 2$_{\varepsilon}$
`&LaTeXe;`	LATEX 2$_{\varepsilon}$
`&LaTeX(2e);`	LATEX(2ϵ)
`&LaTeXTeX;`	(LA)TEX
`&(La)TeX;`	(LA)TEX
`&LyX;`	LYX
`&Metafont;`	METAFONT
`&MetaFont;`	METAFONT
`&Metapost;`	METAPOST
`&MetaPost;`	METAPOST
`&MiKTeX;`	MiKTEX
`&MikTeX;`	MiKTEX
`&PicTeX;`	PICTEX
`&PiCTeX;`	PICTEX
`&teTeX;`	teTEX
`&TeX;`	TEX
`&TeXLaTeX;`	(LA)TEX
`&TeXLive;`	TEX Live
`&TeXlive;`	TEX Live
`&XeLaTeX;`	XƎLATEX
`&Xe(La)TeX;`	XƎ(LA)TEX
`&XeTeX;`	XƎTEX

Table 4.1.: TEX logos

The contents is in principal HTML. In addition some variables can be contained. They are expanded when rendering the file contents.

The same HTML entities are transformed to TeX logos as shown in table 4.1.

4.4. Internationalisation of the search

In the directory `src/main/resources/i18n` there are properties used to prepare the search results. You copy one of the existing properties files:

```
cp search_en.properties search_nl.properties
```

Then you you change the values in this new file.

4.5. Internationalisation of the catalogue data

The catalogue contains some texts which are language specific. They are stored in the database. Those texts have to be translated. The following classes are involved:

PkgCaption

PkgDescription

TopicDetail

5. Environments

5.1. Development environment

The development environment should be located on the personal server of the developers.

missing

5.2. Development environment setup

The development environment should be located on the personal server of the developers.

missing

5.3. Setup a development environment

There are several possibilities to set up a development environment. Here we assume a UNIX-like operating system with a command line shell. The target is a Debian system. Thus it is preferrable to use a Debian system for the development aswell.

5.4. Workspaces

5.4.1. Git

Git is used as version control system for the CTAN site. There are several clients which can be used with Git. In general we decribe the generic command line tool.

To get started you should install Git on your development server. In addition you should have ssh installed.

We have separated the sources into the following repositories:

`https://gitlab.com/comprehensive-tex-archive-network/ctan-site`
: contains the sources for the server and the client.

`https://gitlab.com/comprehensive-tex-archive-network/ctan-content`
: contains the content of the pages – excluding the Catalogue data.

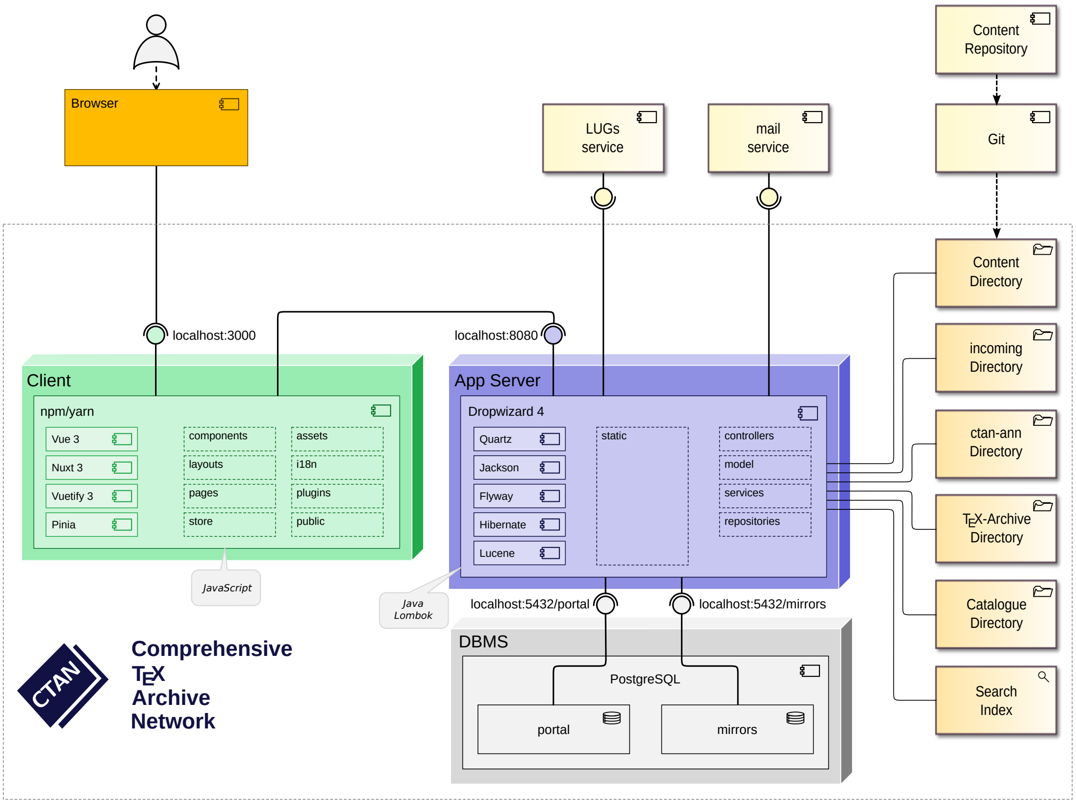


Figure 5.1.: The development environment

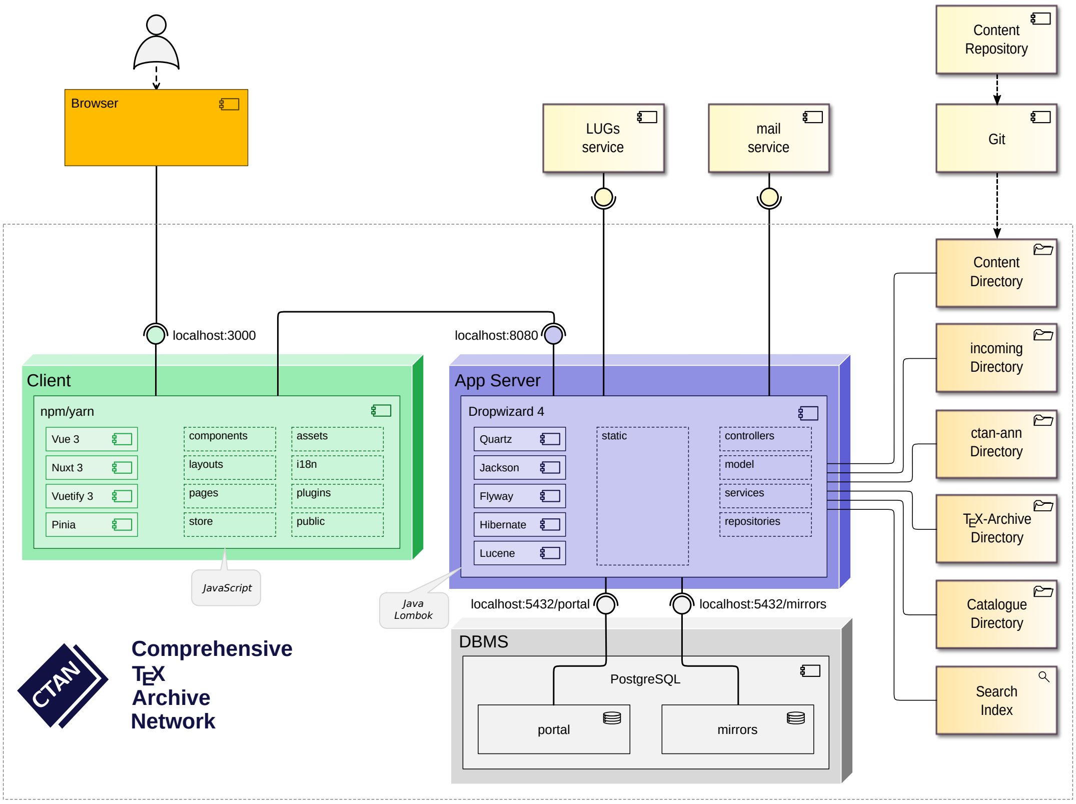
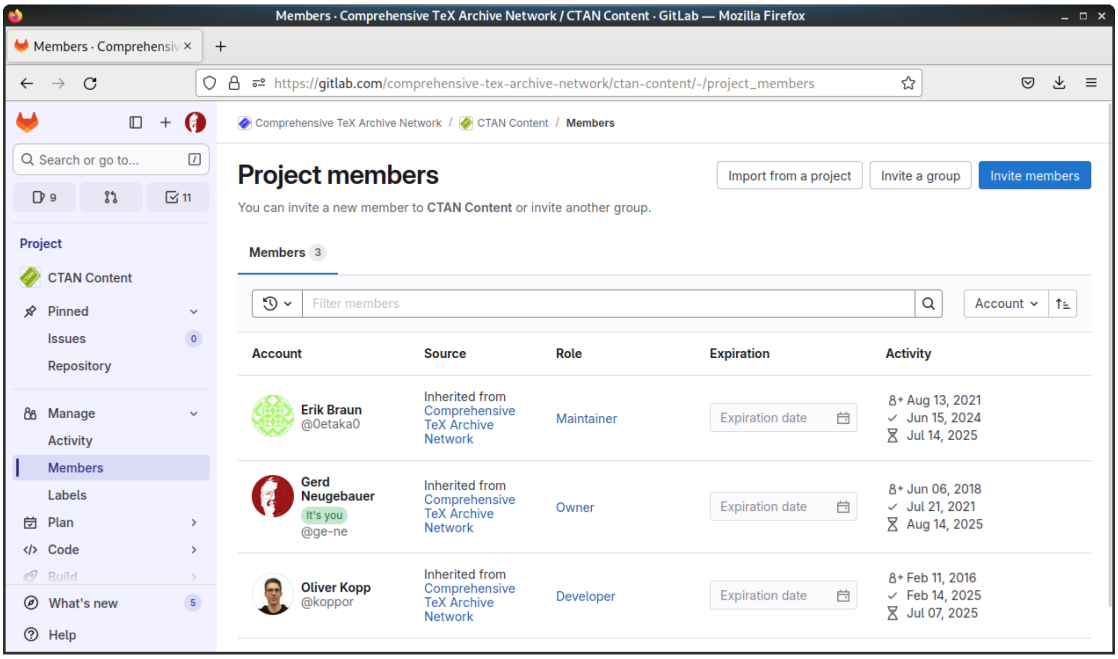


Figure 5.2.: The development environment


Members · Comprehensive TeX Archive Network / CTAN Content · GitLab — Mozilla Firefox
Members · Comprehensiv ×
https://gitlab.com/comprehensive-tex-archive-network/ctan-content/-/project_members
Search or go to...
9
11
Project
CTAN Content
Pinned
Issues 0
Repository
Manage
Activity
Members
Labels
Plan
Code
Build
What's new 5
Help
Comprehensive TeX Archive Network / CTAN Content / Members
Import from a project
Invite a group
Invite members
Members 3
Filter members
Account

Project members

You can invite a new member to **CTAN Content** or invite another group.

Account	Source	Role	Expiration	Activity
Erik Braun @0etaka0	Inherited from Comprehensive TeX Archive Network	Maintainer	Expiration date	Aug 13, 2021 Jun 15, 2024 Jul 14, 2025
Gerd Neugebauer It's you @ge-ne	Inherited from Comprehensive TeX Archive Network	Owner	Expiration date	Jun 06, 2018 Jul 21, 2021 Aug 14, 2025
Oliver Kopp @koppor	Inherited from Comprehensive TeX Archive Network	Developer	Expiration date	Feb 11, 2016 Feb 14, 2025 Jul 07, 2025

Figure 5.3.: The Git project members

Prerequisites

To commit changes to the central repository you need an account on GitLab and have the the role “developer´ ´ (see figure 5.3).

Clone Sources from Git

```
-> cd $ctan-site
-> git clone git@gitlab.com:comprehensive-tex-archive-network/ctan-site.git
```

Commit Sources to Git

Clone Content from Git

```
-> cd $ctan-content
-> git clone git@gitlab.com:comprehensive-tex-archive-network/ctan-content.git
```

Commit Content to Git

5.4.2. Directories

5.4.3. PostgreSQL

The CTAN site requires a database management system (DBMS) underneath. The system PostgreSQL is used for this purpose.

5.4.4. Java

Java at least in version 21 is assumed to be installed and on the path of executables. We suggest to use OpenJDK.

The following command shows a check of the currently installed version:

```
-> java -version
openjdk version "21.0.1" 2023-10-17
OpenJDK Runtime Environment (build 21.0.1+12-29)
OpenJDK 64-Bit Server VM (build 21.0.1+12-29, mixed mode, sharing)
```

5.4.5. Gradle

5.4.6. Node and Yarn

The client part of the software is based on JavaScript. The build system is based on Node and Yarn. Thus these programs have to be installed and available on the path.

Node

Check the installed version:

```
-> node -v
v22.14.0
```

Yarn

Install Yarn:

```
npm install --global yarn
```

Check the installed version:

```
-> yarn -v
4.9.2
```

Jest

Install Jest:

```
yarn add --dev jest
```

Check the installed version:

```
-> yarn jest --version
30.1.3
```

5.4.7. Perl

5.4.8. IDE: Eclipse

Several applications can be used as IDE. Here we describe Eclipse.

5.4.9. make

For the documentation the program `make` is used. For instance the implementation GNU make can be used:

```
-> make -version
GNU Make 4.2.1
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

5.5. LaTeX

For the documentation LaTeX and friends are used. The following programs should be present on the executable search path:

`xelatex` is the executable for XeLaTeX.

`bibtex` is the execitable for BibTeX.

`makeindex` is the execitable for makeindex.

A good choice is to use TeXLive.

5.5.1. opensans

```
...
```

5.6. Production environment

The production environment is mainly located on the server `irony`.

missing

6. Installation of Production

This chapter describes all steps necessary to bring up the CTAN site.

6.1. Prerequisites

In the following description we assume a Debian-based system to be present.

Apache HTTPD should be installed with a package manager in Debian.

Apache Tomcat should be installed with a package manager in Debian.

PostgreSQL should be installed with a package manager in Debian.

Java 21 should be installed with a package manager in Debian.

6.2. Access to the Sources

For simplicity we assume that the sources for the CTAN site have been checked out from the Git repository. See section 5.4.1 for details.

In the following we assume that the location of the directory is stored in the environment variable SITE_SRC:

```
-> export SITE_SRC=`pwd`
```

This is only for simplifying the description of the installation and not required for building or running the site.

6.3. Cron Jobs

Usually the time-based services are run from within the CTAN site as Quartz jobs. The are distributed within the war archive of the CTAN site and need no special installation.

Nevertheless one task is performed by a traditional cron job. This is the checkout of the Catalogue from the Subversion repository. This cron job has to be run by an authorized user. It runs unattended.

6.4. Preparing the Databases

The CTAN site needs a PostgreSQL database installed. The site needs access to two databases *site* and *ctan*.

6.4.1. Creating the Database `site`

The database *site* is used to store the sites own data. Thus the CTAN site needs reading and writing access. For the access to the database the technical user `www` is used.

6.4.2. Creating the Database `ctan`

The database *ctan* contains the information on the CTAN mirrors. The CTAN site just needs reading access to this database. For the access to the database the technical user `www` is used.

If this has not done before use psql:

```
-> psql ctan
```

and grant the required permissions:

```
grant select on all tables in schema ctan to www;
```

Since there is no distinction for the environments only one database is used for all environments.

6.5. Preparing the Working Directory

The CTAN site is mainly run as Web application in Tomcat. This instance of Tomcat is run under the user `tomcat` and the group `tomcat`. We have to make sure that this user or this group have the appropriate reading and writing permissions.

6.5.1. The Index Directory

The search stores the index information in the file system. For this purpose several directories are required. These directories need to be readable by the user under which the servlet container is running. This user is named `tomcat`. The setup is performed with the following commands:

```
-> mkdir -p
-> chgrp tomcat
-> chmod g+rw
```

To set up the server for the test environment the following commands are required:

```
-> mkdir -p -test
-> chgrp tomcat -test
-> chmod g+rw -test
```

And finally in the development environment another set of similar commands does the same:

```
-> mkdir -p -dev
-> chgrp tomcat -dev
-> chmod g+rw -dev
```

6.5.2. The Static Index

The search index contains entries for the static pages as well as for the data from the databases and possibly the TEX archive. Thus it is necessary to populate the index for the static pages.

This command creates and fills the directory . If you run it on the target machine then everything is prepared for the production environment.

If you run it on another machine then you can simply move it to the target machine.

If you are about to prepare the dev of test environment then simply copy or move this directory to the environment's sub-directory.

6.5.3. The Logs Directory

The CTAN site is run in an Apache Tomcat as servlet container. Usually the user running it is named `tomcat`. He belongs to the group `tomcat`. Thus this user needs write access for the places where the log files are located.

```
-> mkdir -p /logs
-> chgrp tomcat /logs
-> chmod g+rwx /logs
```

6.6. Preparing the Web Server

It is assumed that the Apache Web server is used on the frontend. This Web server has to be configured. The configuration must be places in the file

    /etc/apache2/sites-available/www.ctan.org

This file can be found in the sources under

    $SITE_SRC/src/apache2/www.ctan.org

Thus

```
-> cp $SITE_SRC/src/apache2/www.ctan.org /etc/apache2/sites-available/www.ctan.org
```

Some artifacts are served from the Web server. Those are required for producing an error page when the servlet container Tomcat is not reachable. Those artifacts are located in the directory `/htdocs` and provided in the sources on the directory `$SITE_SRC/src/apache2/htdocs`. Thus

```
-> cp -r $SITE_SRC/src/apache2/htdocs /serv/www/www.ctan.org/
```

6.7. Building the Web Application

To build the war file of the CTAN site follow the steps described in chapter 7. Finally you should have a file named `target/ctan-site-3.*.war` to continue with.

6.8. Deploying the Web Application

We assume that the war for the CTAN site has been created (see section 7). The war should be named `ctan-site-x.y.z.war` where *x.y.z* is the version number of the current build.

First, we need a ssh tunnel to the production host:

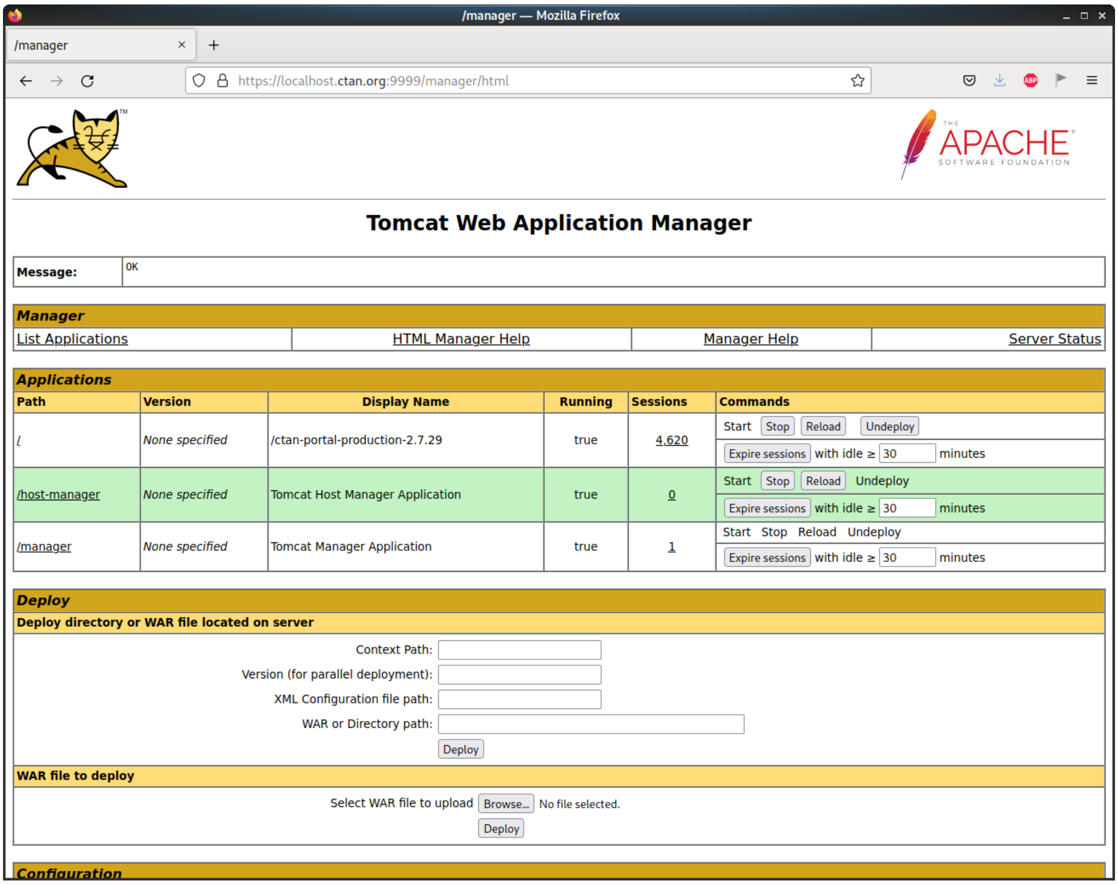
```
-> ssh irony -f -L 9999:localhost:443 -N
```

If Tomcat has the management Web application installed then you can navigate to `http://localhost.ctan.org:9999/manager/html/`. Here you find a list of deployed web applications (see figure 6.1)

1. Create a lock file for the maintenance. The creation of this lock file redirects users to the maintenance page by the Apache. No traffic is passed to Tomcat.

   ```
   -> touch /serv/www/www.ctan.org/maint.lock
   ```

2. Next undeploy any instance already running.
3. Then you can deploy the war file under the context `/ctan-site` (see figure 6.2).
4. Restart Tomcat to prevent problems with the memory management:


/manager — Mozilla Firefox
/manager
https://localhost.ctan.org:9999/manager/html
THE APACHE SOFTWARE FOUNDATION

Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	*None specified*	/ctan-portal-production-2.7.29	true	4,620	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	*None specified*	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	*None specified*	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

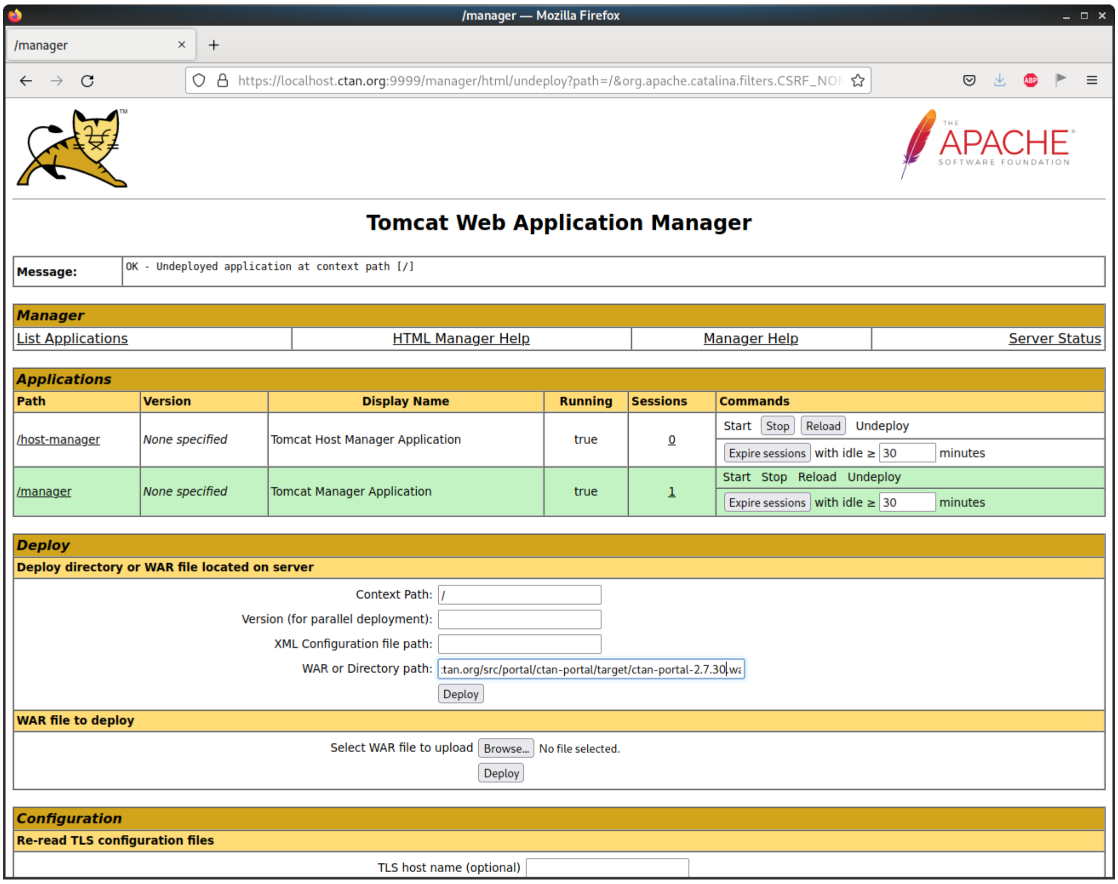
Context Path:
Version (for parallel deployment):
XML Configuration file path:
WAR or Directory path:
Deploy

WAR file to deploy

Select WAR file to upload Browse... No file selected.
Deploy

Configuration

Figure 6.1.: Tomcat manager


/manager — Mozilla Firefox
/manager
https://localhost.ctan.org:9999/manager/html/undeploy?path=/&org.apache.catalina.filters.CSRF_NO
APACHE SOFTWARE FOUNDATION

Tomcat Web Application Manager

Message:	OK - Undeployed application at context path [/]

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/host-manager	*None specified*	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	*None specified*	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path:	/
Version (for parallel deployment):	
XML Configuration file path:	
WAR or Directory path:	:tan.org/src/portal/ctan-portal/target/ctan-portal-2.7.30.wa

Deploy

WAR file to deploy

Select WAR file to upload Browse... No file selected.

Deploy

Configuration

Re-read TLS configuration files

TLS host name (optional)

Figure 6.2.: Deploying a Web application

```
-> sudo /etc/init.d/tomcat-ctan restart
```

5. Release the lock to reactivate the site:

```
-> rm /serv/www/www.ctan.org/maint.lock
```

6.9. Supporting Utilities

The files contained in the sources under `$SITE_SRC/bin` contains a supporting utility. It should be copied into the directory `/bin`.

Beware to preserve the permission bits for the executable.

7. Build

7.1. The build server

The build server is located on build.ctan.org (aka melody.ctan.org).

7.1.1. The build job

The build job is a cron job which runs the program

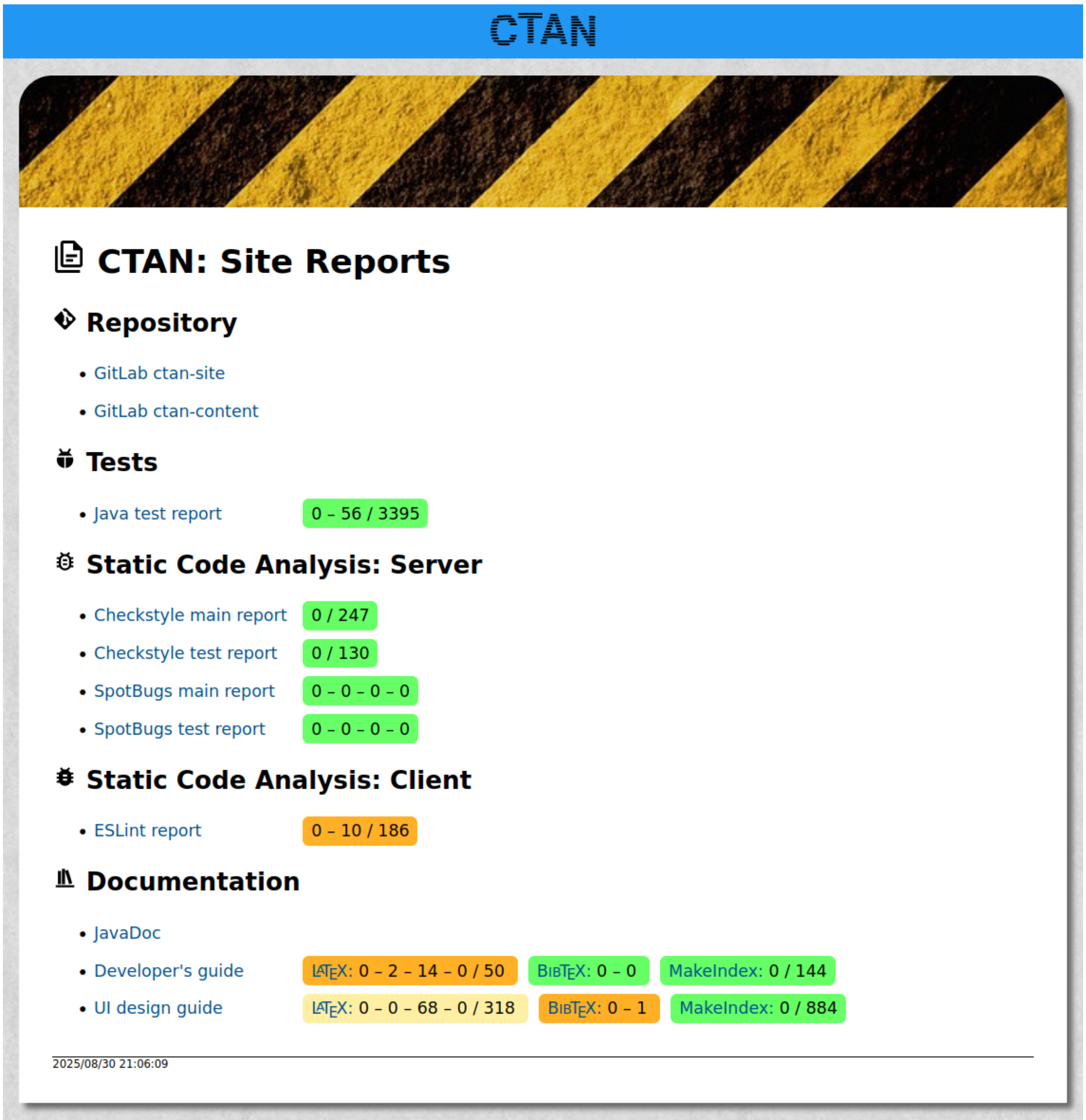
`/serv/build/ctan-site/build.ctan.org/generate.sh`

The build job generates a report under `https://build.ctan.org` as HTML page. An example screenshot is shown in figure 7.1.

7.2. Build manually

The building is mainly performed with the help of Gradle. We can start it in the main directory of the workspace.

```
-> gradlew
```



CTAN: Site Reports

Repository

- GitLab ctan-site
- GitLab ctan-content

Tests

Java test report	0 – 56 / 3395

Static Code Analysis: Server

Checkstyle main report	0 / 247
Checkstyle test report	0 / 130
SpotBugs main report	0 – 0 – 0 – 0
SpotBugs test report	0 – 0 – 0 – 0

Static Code Analysis: Client

ESLint report	0 – 10 / 186

Documentation

JavaDoc			
Developer's guide	LaTeX: 0 – 2 – 14 – 0 / 50	BibTeX: 0 – 0	MakeIndex: 0 / 144
UI design guide	LaTeX: 0 – 0 – 68 – 0 / 318	BibTeX: 0 – 1	MakeIndex: 0 / 884

2025/08/30 21:06:09

Figure 7.1.: The build reports

8. Quality assurance

8.1. Java: unit tests

8.1.1. Definition of the unit tests

The unit tests are written utilizing JUnit.

The unit tests can be found in the directory `src/test/java`.

8.1.2. Running the unit tests

To run the unit tests issue the following command:

```
-> ./grailsw test
```

The results can be found in the directory `build/reports/tests/test`.

8.2. Java: Checkstyle

Checkstyle is a tool for static code analysis for Java code. It is focusing on uniform appearance of the code. It has a larger set of rules which are checked and might lead to errors or warnings.

8.2.1. Definition of the Checkstyle rules

The file `config/checkstyle/checkstyle.xml` defines the Checkstyle rules used in this project.

8.2.2. Running Checkstyle

To run Checkstyle on the productive Java code issue the following command:

```
-> ./grailsw checkstyleMain
```

The results can be found in the directory `build/reports/checkstyle/main.html`.

To run Checkstyle on the Java test code issue the following command:

```
-> ./grailsw checkstyleTest
```

The results can be found in the directory build/reports/checkstyle/test.html.

8.3. Java: SpotBugs

SpotBugs is a tool for static code analysis of Java code. It has a larger set of rules which are checked and might lead to errors or warnings.

8.3.1. Running SpotBugs

To run SpotBugs on the productive Java code issue the following command:

```
-> ./grailsw spotbugsMain
```

The results can be found in the directory build/reports/spotbugs/main.

To run SpotBugs on the Java test code issue the following command:

```
-> ./grailsw spotbugsTest
```

The results can be found in the directory build/reports/spotbugs/test.

8.4. Vue: ESLint

ESLint is a tool to analyse JavaScript code and check a set of rules. We use it to detect errors and improve the uniformity of the code.

8.4.1. Definition of the ESLint rules

The rules applied for this project can be found in the file

src/client/eslint.config.js.

8.4.2. Running ESLint

To run ESLint on the client code issue the following command while you are in the directory src/client:

```
-> yarn lint
```

Alternatively you can use the following command while you are in the root directory of the workspace:

```
 -> ./gradlew eslint
```

Appendix

A. Glossary

Author

See Contributor.

Catalogue

The Catalogue is the source of all information about packages, authors, and topics on CTAN. It has been created initially by Graham J. Williams and is now maintained by the CTAN team.

The Catalogue is stored in a set of XML files. Initially static Web pages have been generated from these sources. Nowadays the XML file are regularily imported into a database. From there the data is presented on the Web.

CDN

Content Delivery Network.

Contributor

A contributor is a person or group who are related to a package. This can be as active or former maintainer or as uploader.

Formerly the contributor has been called "author".

The identifier for a contributor is assigned by the CTAN team. It usually consists of the family name in lower-case. Optionally the initial of the first name is appended after a minus sign for disambiguation.

CTAN

CTAN is the name of the Comprehensive TEX Archive Network. Initially it consisted of three primary servers in Germany, the US, and the UK and a lot of mirrors. Nowadays the internet is more performant and more reliable. Also the maintainers in US and UK have left the team. Thus only one primary server in Germany is left behind.

CTAN-ann

CTAN-ann is a mailing list to publish the announcements of new and updated packages on CTAN. The email address is ctan-ann@ctan.org.

Incoming

The uploaded packages are stored in the incoming directory. The incoming directory is `/serv/www/www.ctan.org/incoming/`. The upload managers take care of updating the catalogue and move the content to the TEX-archive.

License

The packages on CTAN are provided with the licenses as defined by the contributors. These can be Open-Source licenses but are not restricted to those. The TEX distributions which take the packages from CTAN may be more restrictive and offer only free packages.

The software of the CTAN site itself is distributed under the 3-clause BSD license (see page 61).

LUG

LUG is an abbreviation for local user group. user groups for TEX, LATEX, and friends have gathered around the world. They a reorganised in local user groups.

MiKTEX

MiKTEX is one of the major TEX distributions. It gets its contents directly from CTAN.

MiKTEX is present on CTAN in https://ctan.org/pkg/miktex.

Mirror

Mirrors are secondary servers around the world which provide access to the TEX archive. For this purpose they copy the content of the primary server and offer it via HTTPS mand maybe HTTP and FTP.

In order to be an official mirror the server has to be registered at CTAN.

Mirror database

The mirrors are kept in a separate database. This mirrors database is read by the CTAN site. The update is external.

Mirror monitor

The mirrors are monitored by CTAN. This allows us to redirect the users only to the reachable and up-to-date mirrors.

Package

A package on CTAN denotes a set of files to achieve a certain goal when typesetting documents with TEX and friends.

Package alias

A package may have alternative ids. If an alias is used in the package URL then the browser is redirected to the primary id.

Package id

A package on CTAN is uniquely identified by the id attribute. The package id is composed of lower-case characters, digits, the minus sign, and in rare cases the underscore.

The key is used in the URL for the package: `/pkg/«key»`

Package name

A package on CTAN is identified by a name. The name is used in the title of the package. If no package name is specified then the package id is used instead.

TeX archive

The TeX archive is a directory structure which contains the sources of the contributed packages on CTAN. The top level directory structure vontains the following directories:

- biblio
- dviware
- fonts
- graphics
- help
- indexing
- info
- install
- language
- macros
- obsolete
- support
- systems
- tds
- usergrps
- web

TeX Live

TeX Live is one of the major TeX distributions. It gets its contents directly from CTAN.

TeX Live is present on CTAN in https://ctan.org/pkg/texlive.

Topic

Topics provide a means to classify the packages on CTAN. Initially the directory structure of the TEX archive was the only way to express a classification. The topics allow more expressiveness since a package may have more than one topic.

Uploader

The uploader is the person who uploads a package to CTAN. Usually the uploader is the author. The uploader can be a different person. In this case the right to upload a package has been given to this person and recorded on CTAN.

B. BSD 3-Clause License

References

[Lam86] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 2nd edition, 1986.

[RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual*. Pearson Higher Education, 2nd edition, 2004.

[SCR] The 2020 Scrum Guide. https://scrumguides.org/scrum-guide.html.

[SEM] Semantic versioning 2.0.0. https://semver.org/.

[TA05] Jeff Tyree and Art Akerman. Architecture decisions: Demystifying architecture. *IEEE Software*, 22(2):19–27, 2005.

Index