Karlsruhe Institute of Technology

Broadcasting Webservice for Smart TVs

Lennard Kittner, Henry Boos,
Jiangang Huang, Sicheng Dong,
Jannik Wibker

Contents

1 Purpose

1.1 Product Goal

"Can't this TV display something useful"

Many schools, work places and chairs have large TVs hanging around which can potentially show a lot of useful information. Most of their limitations revolve around poor layout for the given screen size and long waiting times until the wanted content shows up. This can be addressed by displaying multiple things at the same time if the screen real estate allows for it. This way a lot more information can be seen at once and the time spent waiting is reduced.

The goal of this product is to deliver an extensible, easy to configure all-in-one solution that allows creating custom layouts tailored to the smart TV at hand. The product comes with most of the core functionality expected of such a smart TV system out-of-the-box and its easy extensibility makes creating additional custom tailored widgets and plugins easy.

1.2 Target Audience

The target audience of this product is departments which have the appropriate hardware available, or could easily acquire it, and want to display meaningful information to visitors, coworkers, students or whoever else might come across the smart TV .

2 Scenarios

2.1 A normal work day

For this scenario it is assumed that the following optional requirements are implemented: FRO-1 (Weather widget), FRO-4 (Tram / bus schedule).

Marc is an employee at the Chair of Embedded Systems. In the morning he gets into the building, first thing he does is take a look at the smart TV in the hallway to see if there are any meetings scheduled for the morning. Next time he passes by the smart TV while fetching himself a coffee, he once again takes a look to see which one of the servers has the lowest utilization, so he knows which one to use for his project. Later that day Marc and his other coworkers meet in the hallway, they all take a quick look at the smart TV and discuss what they want to eat in the mensa today. After returning from lunch and not seeing any new announcements or meetings, he continues with his work. With his work finished for the day Marc again checks the smart TV on his way out to know the current weather situation and wether his tram has any delays or not.

2.2 Notifying colleagues

Tina, also working at Chair of Embedded Systems, gets an e-mail about an upcoming internet outage. To inform the other people in the building the she opens the dashboard on her computer, logs into the dashboard using the IDP and creates a new announcement starting now and ending at the end of the outage. After doing that Tina gets another e-mail telling her about an upcoming presentation about RISC-V, because of which she decides to add a new calendar event to the calendar once again using the dashboard to remind everyone.

2.3 A busy work day

For this scenario it is assumed that the following optional requirements are implemented: FRO-1 (Weather widget), FRO-2 (Room usage widget), FRO-4 (Tram / bus widget).

Sam works at the Chair of Embedded Systems. As Sam enters the building and goes to the office while passing by the smart TV. There Sam notices an important event in the calendar. After some work Sam walks to the event that was displayed earlier. On the way Sam passes by the smart TV again and takes a look at todays mensa menu. Once the important event has come to an end the attendees decide to split up. Sam opens the smart TV website to look up which of the meeting rooms is currently empty. After that it is lunch break. On the way out of the office Sam sees the weather widget on the smart TV and notices that it will start raining on the way back from the mensa, so Sam decides to pick up an umbrella. After Sam comes back from lunch the smart TV displays an announcement which says that there will be an internet outage in two hours. Because of that Sam decides to leave early. Before leaving Sam takes one final look at the smart TV. Sam doesn't have to hurry as the smart TV shows the tram as being delayed a few minutes.

3 Overall Description

3.1 Product Environment

3.1.1 Backend Server

- A network connection (optionally an internet connection depending on widgets)
- 2 logical CPU cores
- 4 GiB of RAM
- 10 GiB of storage space

Software requirements

- A 64-bit Operating System
- Node exporter (if the server usage widget (FRO-9) is used) on targeted server
- A recent version of Docker
- A recent version of Docker Compose
- Python 3.6 or later (required for Docker Compose)

The backend will be written in Java and use Spring for network requests.

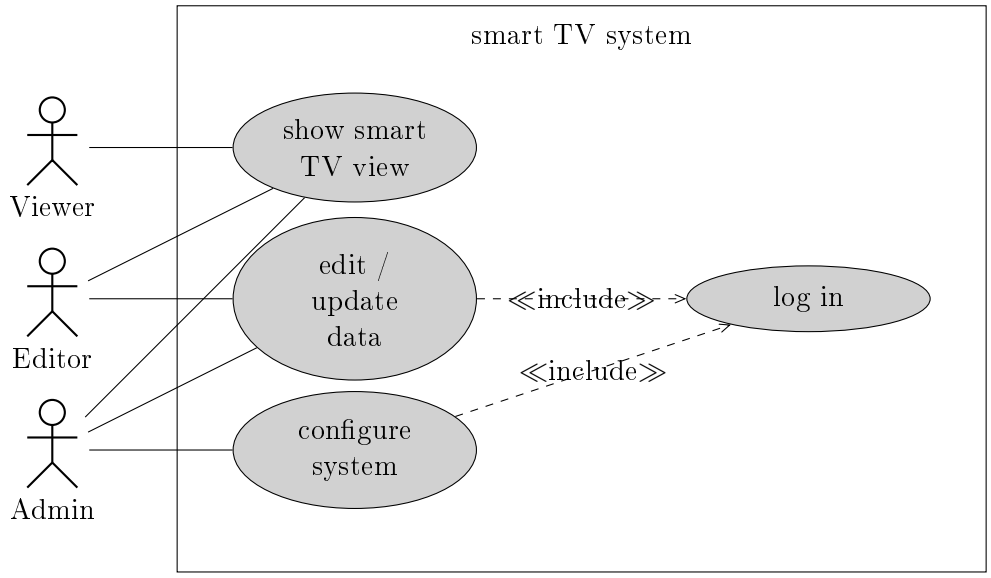
3.1.2 Frontend Application

Hardware & Software requirements

A device with at least 1 GiB of RAM and a 2 Ghz single core processor running a browser capable of executing ES6 code. The browser needs to have support for all browser features introduced to the W3C web platform specifications 2018 (or later).

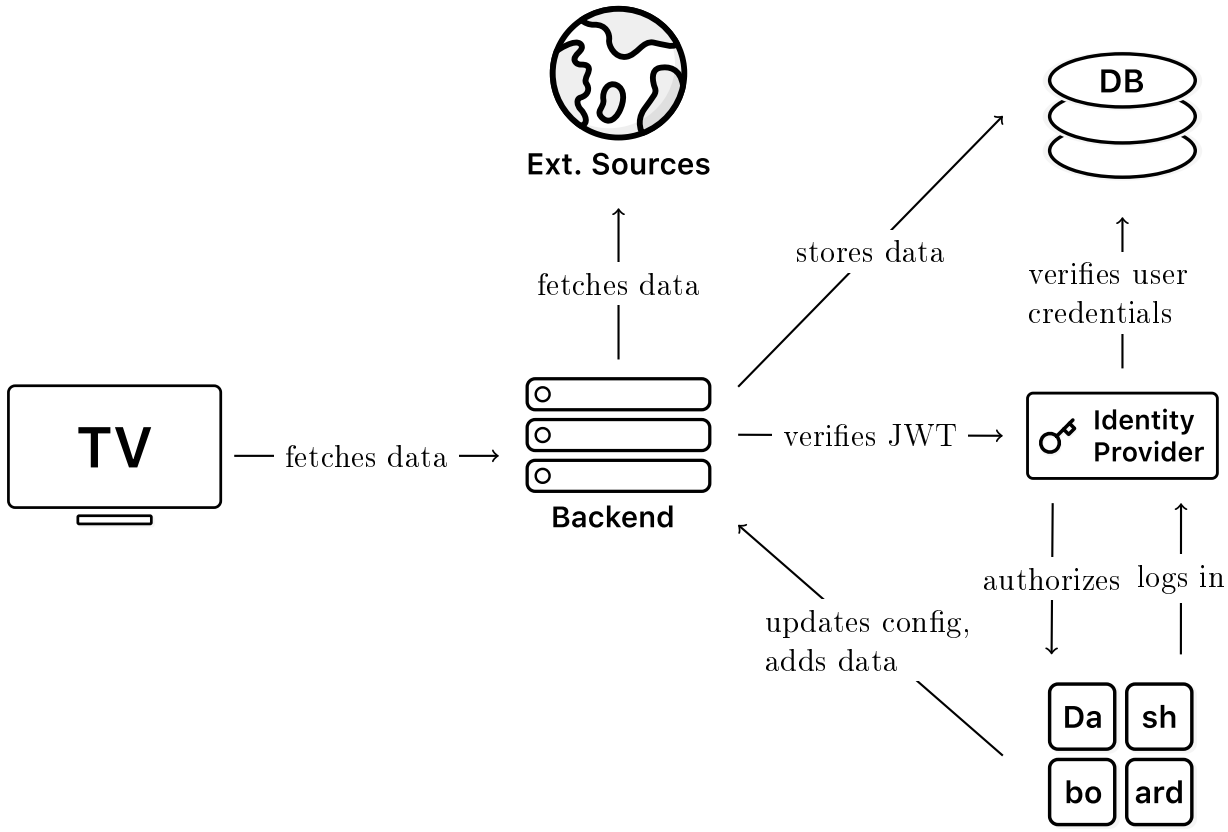
The frontend will be written in Typescript and use React for rendering.

3.2 Use Case Diagram



3.3 System Model

3.3.1 General Concept



The diagram shows the main architecture of the product.

The dashboard (FR-6) communicates with the IDP which in turn has a database for storing credentials. Using the dashboard an admin can update the configuration of the frontend and backend using config files. By updating configuration files the user can for example change the layout shown on the smart TV or add new calendars to be displayed.

Other users such as editors have less permissions, editors can create announcements, upload images to display on the smart TV and manually create calendar events to display.

The backend will save some data of its own in the database such as announcements and manually added calendar events (non-exhaustive list).

The smart TV periodically asks the backend for new data (HTTP polling) to which the server then responds with an aggregate of data collected from external sources such as calendars or RSS feeds.

Plugin systems and config files

The backend stores multiple config files which can be up- and downloaded using the dashboard. These files can also be retrieved by the frontend. The frontend loads some of these on page load.

Plugins exist in the frontend as well as the backend. Plugins in the frontend are not technically the same as widgets but are often referred to as such because no other plugins exist for the frontend as of right now. The idea of a plugin is more general than the idea of a widget, widgets are a subset of plugins.

Multiple functional requirements require some kind of data retrieval and fetching in the backend. This is done through a "companion plugin" in the backend which retrieves the data the widget in the frontend needs. These are often just called "data plugins" as their job is retrieving and aggregating data.

This is however not a one-to-one mapping. One widget may theoretically require data from multiple data plugins and one data plugin might serve multiple widgets.

An example of this are the *Calendar widget* (FR-7) and *Room usage widget* (FRO-2) which both rely on the *Calendar data plugin*.

Since this is not a one-to-one mapping in some instances (it is most of the time however) separate config files also exist for frontend and backend plugins.

4 Stored and Processed Data

Data is mostly stored and processed on the server. The server has a multitude of config files which contain the majority of actually stored data. Additionally a database is being used for some things in the backend. This database is also used by the IDP.

The client retrieves almost all its data from the server, including the config files it needs.

4.1 Backend

4.1.1 Data for mandatory requirements

Plugin system

All (backend) plugins are stored as compiled binaries. Each plugin has one config file available to it. For each plugin a

1. name,
2. unique id and a
3. config file name

need to be configured at minimum, specific API features the backend provides may need additional data. This data is stored inside the plugin itself.

The "Plugins"-config file specifies which plugins need to be loaded.

Account and permission system

The IDP (ref.: General Concept) stores its user account data in the database.

The data includes:

1. A username,
2. a display name,
3. a salt,
4. a hashed and salted password,
5. a role (either “admin”, “editor” or “viewer”)

Internationalization

The backend stores the setting which languages are to be shown in a config file.

The individual translation files are not stored in the backend but rather in the frontend and dashboard.

Calendar

Store the addresses and information needed to access the calendars.

Additionally for each calendar a *name* and a *color* are stored. These can be used to make distinguishing between different calendars in the GUI easier.

Publications

All tracked authors and tracked websites are stored in a config file. For each publication the following information is retrieved:

1. The author or authors involved in the publication.
2. The name of the publication.
3. The date the publication got published.
4. A short description of the publication (if available).
5. The place where the publication was published (if available).

Announcement

The announcement text, the urgency as well as its start and expire time is saved in a database.

Error logs

If something triggers the generation of a warning or error this warning or error is saved to a log file which can be seen in the dashboard (FR-11 and FR-6). This log file is stored on the server.

Each warning is a *message* string prepended with a *timestamp* (ISO-8601) and an *urgency*.

Cafeteria dishes

The *Cafeteria data plugin* (FR-12) retrieves the cafeteria menu from the website. The default implementation of the plugin scrapes the Studierendenwerk Cafeteria website for information but this can be changed by updating the plugin.

The following data is extracted:

1. Name of meal
2. Price of meal
3. Allergy & food intolerance related data (if available)
4. Wether it is vegan / vegetarian or contains meat (if available)
5. Type of meat if meal contains meat (if available)
6. Location where the meal can be bought (e.g. “line 1”)

4.1.2 Data for optional requirements

Weather

A list of locations for which weather data is to be displayed is saved in a config file.

For each such location the weather information is retrieved using an API. How this API looks exactly is not important as it is going to be handled by the *Weather data plugin*. The API will most likely also need some kind of API key, this is also saved in the *Weather data plugin* config file.

Printer queue

All the printers and a way to access their printer queue are stored. All the print jobs in the printer queue are saved. This only includes a *name*, an *id* and the *person* who started the print job (if available).

Room usage

A list of rooms is stored in a config file. For each room a *primary display name* and multiple *alternative names* can be stored. When searching for free and occupied rooms using calendar events all of the names will be considered when determining the status of a room. This widget uses the *Calendar data plugin* for its information.

RSS feed

Each RSS feed is stored in a config file using its *URL* and a *name* associated with it.

When processing the XML data the following is extracted:

A "Channel"-element used to describe the RSS feed:

1. Title
2. Link
3. Description
4. Items
    - Title
    - Link
    - Description

Tram / bus schedule

The *Tram & bus schedule data plugin* is responsible for retrieving all tram and bus related information. It uses a config file for configuration.

The default implementation of the plugin queries data from the Karlsruher Verkehrsverbund and Deutsche Bahn.

The queried data is then transformed to a useful data format which (for each itinerary and journey) includes:

1. type of transportation
2. name of itinerary
3. time of departure
4. time of arrival
5. possible delay at each station
6. starting and ending station

Image

Albums are saved. Each album consists of multiple *images* and a *name*. An image is represented by an *URL* pointing to the image. This information is saved in a database.

Server usage

Node exporter is used to retrieve the server usage of tracked servers.

Processed server usage data includes:

1. Server name and id
2. CPU utilization
3. RAM usage
4. Network throughput

4.2 Frontend

The frontend does not permanently store any data. It only processes and temporarily stores data on runtime. Some data may be bundled with the frontend however.

Generally the frontend consumes all the aggregated data by the backend. Only additionally processed data is listed here.

4.2.1 Data for mandatory requirements

Layout system

The client loads the "Layout"-config file from the server and parses it. This file describes the layout the smart TV will display.

It consists of a recursive data structure describing how the "root slot" is split into smaller slots according to FR-1 and which widget is displayed when, for how long and with what properties. These properties are small configuration options passed to the instances of widgets (e.g. C° / F° for the *Weather widget*). Since these are instance-local different instances can have different properties.

Widget and plugin system

Widgets are stored as transpiled javascript files which are loaded at runtime or come already bundled with the core frontend code.

Each widget has

1. a name,
2. a unique id and
3. a config file location

associated with it.

Each widgets config file can globally configure some properties of the widget for all its instances. These properties can however be overwritten by properties passed through the layout system to the exact instance as described in the Layout system.

Internationalization

The frontend has a set of translation files which contain the translations for each individual UI element. The translations are stored as key-value pairs.

Additionally the frontend also has files containing other internationalization related data such as date format strings.

The frontend also has a list of languages it should display. This is retrieved from a config file the client loads from the server.

Clock

A list of timezones is stored in a config file. The layout specifies for each instance of the widget which timezone is to be used. A default for the 12 hour / 24 hour time display setting can be configured, this can be overwritten using the Layout system.

Error logs

The frontend may generate some errors which are then sent to the server to be saved in a log file (FR-11).

4.2.2 Data for optional requirements

Weather widget

The config file for the *Weather widget* specifies a list of locations. The file also specifies a global setting for each *Weather widget* wether to use Celsius or Fahrenheit.
This can be overwritten on an instance by instance basis using the Layout system.

Themeability

Themes are stored as CSS files. The currently active theme is stored in a config file which retrieved and processed by the frontend. If the theme should change based on the time of day this is also stored in a config file.

4.3 Dashboard

The dashboard up- and downloads all the config files. When uploading it checks if the syntax of the config file is correct but does no further processing on it.

4.3.1 Data for mandatory requirements

Internationalization

The dashboard extracts the configured language from the "General"-config file and displays the UI in that language. The dashboard has translation files bundled with it, same as the frontend.

Error logs

The dashboard displays the error log (FR-11) and parses the string of each entry into 3 parts. The *urgency*, the *date and time* it happened and the *message*.

4.3.2 Data for optional requirements

Themeability

The dashboard extracts the configured theme from the "General"-config file and displays the UI in that theme. The dashboard has CSS files which describe each theme, same as the frontend.

5 Specific Requirements

5.1 Functional Requirements

5.1.1 Mandatory Requirements

FR-1 Layout system

Goal: Have a customizable layout system that fits allows fitting the layout to the specific use case of the smart TVs.

Precondition: A layout is configured or a default configuration has been selected.

Postcondition (success): The configured layout is used.

Postcondition (fail): If nothing is configured the *root slot* acts like an *empty slot*.

Description: The layout of the smart TV is divided into slots. A slot can house a widget, can be split either horizontally or vertically into two more slots, or contain a list of slots to cycle through. There are therefore multiple types of slots:

- *widget slot*: Houses a single widget.
- *vertical (split) slot*: Slot which is divided in a lower part and an upper part.
- *horizontal (split) slot*: Slot which is divided into a left part and a right part.
- *carousel slot*: A slot which houses multiple slots (may be referred to as sub-slots). Only one sub-slot is shown at a time and after a configured time (defaults to 15 seconds) the displayed sub-slot is switched out for the next sub-slot in the list of sub-slots. If th end of the list is reached the list wraps around to the start. A visual indicator may or may not be overlayed on top of the current sub-slot indicating the current position in the list of sub-slots.

- *empty slot*: A placeholder slot which does not display anything except hte background color choosen by the theme.

For *vertical* and *horizontal split slots* the following applies:
Each of the two parts houses another slot (may be referred to as a sub-slot) which can be of any kind. The position where the split happens is configurable and defaults to 50% of the slots space for each sub-slot. Values are given as percentages and all the space needed for visual dividers between the slots is subtracted from the total space the slot has.

An extra slot called the *root slot* exists which houses all other slots. If no layout is defined it acts like an empty slot.

FR-2 Plugin system

Goal: Make extending the functionality of the smart TV easy without making modifications to the core codebase.

Precondition: Plugin exists, is in the list of plugins and in the list of activated plugins in the configuration.

Postcondition (success): Plugin is loaded successfully.

Postcondition (fail): Plugin failed to load and an appropriate error message is generated (FR-11).

Description: The plugin system allows to extend the functionality of the smart TV beyond the core features. This means an extensive API is supplied for plugins. This plugin system exists for both the frontend and backend. Plugins for each one can exist on their own or require another plugin in the frontend or backend respectively. Most of the functionality of the smart TV is implemented as a set of (core) plugins.

FR-3 Internationalization system

Goal: Make the smart TV useable by as many people as possible by allowing different languages to be displayed. This can be done at runtime where all text is swapped with a version in another language or at setup where a language is set.

Precondition:

- The language is not RTL. It needs to be strictly LTR.

- All the appropriate files have been created and added. These files contain all translations to a different language.

Postcondition (success): The smart TV displays everything possible in the selected language. Some text which is not part of the core system, like calendar events and other external input might not be available in the selected language.

Postcondition (fail): If anything fails the smart TV falls back to displaying all UI elements in english, as english is the default language and it is guaranteed that all translations are present, an error is generated according to FR-11.

Triggering Events:

- The configuration specifies a specific language *or*
- The smart TV is set up to cycle through different languages

Description: The smart TV should display as much content as possible in the selected language. A language can be specified for calendars. If multiple versions of the same calendar exist with different languages only the appropriate events are shown. If only one version of a given calendar exists it is displayed irregardless of its language. The smart TV can cycle through multiple different languages if they are configured and set up correctly. The language includes formatting for dates and time as well as possibly numerals.

The language system is a simple representation of the real world. Complex structures such as languages with different writing systems (this includes e.g. British/American English or Simplified/Traditional Chinese) would be counted as completely separate languages and some duplication between the translations might be required.

FR-4 Time-configurable widget display

Goal: Show widgets only when they are necessary.

Precondition: An admin has configured time slots for specific widgets in which they are to be shown.

Postcondition (success): The widget is only shown in a certain time period.

Postcondition (fail): The configuration is ignored and therefore the widget is shown all the time.

Triggering Events: An admin has configured a widget to only be shown at a certain time period.

Description: Some widgets are only relevant sometimes. This feature allows specifying multiple periods of time in which a given widget is to be shown. This might be used to only display e.g. cafeteria dishes (FR-12) shortly before the cafeteria opens.

FR-5 Permission system

Goal: Restrict access of some features to specific roles. These roles are: *Admin*, *Editor* and *Viewer*.

Precondition: The initial setup is complete and an IDP is specified.

Postcondition (success): A user can only access features their role has permissions to.

Postcondition (fail): Instead of successfully logging in an error page is displayed.

Triggering Events: A login attempt has just been made.

Description: The permission system ensures that

1. only *admins* can make changes to the configuration or have access to some features *and*
2. *admins* and *editors* can make changes to the content displayed displayed on the smart TV *and*
3. viewers are able to see the view that is otherwise only visible on the smart TV.

FR-6 Dashboard for configuration and data entry

Goal: Give admins a way to set up the smart TV and change the configuration options. Give editors a way to enter announcements (FR-9) and calendar events (FR-7) as well as images to display (FRO-8) manually.

Postcondition (success): The admin updated the configuration of the smart TV. The smart TV and the backend will then update themself accordingly.

Postcondition (fail): Updating the configuration failed and an appropriate error message is given. The change will be reverted and the system restarted if the change requires it.

Triggering Events:

1. An *admin* opens the dashboard and logs in.
2. The *admin* changes the configuration
3. The *admin* hits a button to save and apply the new configuration.

Description: The dashboard allows authorized users to do the initial setup of the smart TV as well as changing the configuration of the smart TV later on. Depending on the change a restart of the backend and/or frontend may be required. The system will be able to do this itself if necessary. Changing the configuration will be done by uploading new config files. There are multiple configuration files for different aspects of the system:

- *General config*: Contains configuration settings for timezone, locale(s), currently selected theme, frontend polling rate (non-exhaustive list)
- *Plugins config*: Contains a list of available plugins and a list of active plugins. Configuration needed for FR-4 will also be placed in this file.
- *Layout config*: Contains configuration for FR-1.
- *Other*: Each plugin has its own config file which is named in accordance with the plugin name.

FR-7 Calendar widget

Goal: Display calendar events on the smart TV when they are relevant.

Precondition: One or multiple calendars have been set up and contain events that can be displayed.

Postcondition (success): A list of calendar events is being displayed on the smart TV and is periodically updated.

Postcondition (fail): The list could not be displayed and appropriate error messages are generated (FR-11).

Triggering Events: An admin activates the *Calendar widget* and *Calendar data plugin* through the dashboard.

Description: The smart TV lists upcoming events ordered by date and time. If events are marked as "all day" they will be shown separately. If only a few events are within

a one day period older events may be shown as well, they will however be visually separated from the newer ones. An admin can configure a color which might be used to visually differentiate events from certain calendars. This colors can be set on a per calendar basis. The smart TV might however discard this information and display all events uniformly if space or theme requirements mandate this. The list of events is periodically updated on the client as well as on the server. The time between updates to the frontend is globally configured for all updates to the frontend (frontend polling rate). The interval between calendar fetches in the backend is configurable per calendar. An editor can manually add events through the dashboard, he can specify a name, location and can choose between start and endtime or all day.

FR-8 Publications widget

Goal: Show the latest publications of tracked authors.

Precondition: A list of authors and/or websites to track is configured and at least one publication exists. The websites need to be up and running.

Postcondition (success): The most recent publications of tracked authors / on tracked websites are displayed.

Postcondition (fail): No publications were found or no authors or websites have been added. Instead a message is shown that no publications could be found.

Triggering Events: An admin activates the *Publications widget* and *Publications data plugin* through the dashboard.

Description: Newly published publications of authors can be displayed on the smart TV. The following is shown for each publication:

- Name or title of the publication
- Publishment date
- A short description (if available)
- The place where the publication was published (if available)

FR-9 Announcements

Goal: Be able to display announcements created by editors and admins.

Precondition: An announcement has been created by an admin with a set expire time.

Postcondition (success): The announcement is shown on the smart TV.

Postcondition (fail): An appropriate error message will be generated (FR-11).

Triggering Events: The widget will be shown if an announcement is currently active.

Description: An announcement is a text message. Announcements can be created by editors or admins. It is possible to change the urgency of the announcement which will be reflected by the background color shown on the smart TV. Multiple announcements can be shown at the same time, in this case the display switches between the active announcements. Each announcement has an expire time after which it expires and will be deleted.

FR-10 Clock widget

Goal: Be able to display the current time and if configured the time of other timezones.

Precondition: A default timezone has been configured in the "General" config file. If this is not the case GMT (Greenwich Mean Time) is shown and an error is generated (FR-11).

Postcondition (success): The time is shown in a slot on the smart TV.

Postcondition (fail): An error is generated (FR-11).

Triggering Events: An admin activates the *Clock widget* through the dashboard.

Description: The time is displayed on a slot on the smart TV. If there are multiple timezones configured they are individually named (according to the timezone name). If the default timezone (which one this is is configured in the "General" config file) is shown the name is shown as "Local time (<GMT+offset>)" (or a translated version of that; FR-3) with offset being the appropriate timezone offset for the default timezone. This extra indicator can be omitted by the smart TV if only the default timezone is configured. The smart TV may , depending on current circumstances, decide to display the timezone names even if it the previous requirements did not require it. This can *for example* be used to signify an imminent switch from/to daylight saving time. If configured this way the date can be displayed alongside the time.

If multiple timezones are to be displayed this needs to be configured using the Layout system (FR-1).

FR-11 Error logging

Goal: Logging the errors happening and displaying them in the dashboard as well as saving them in an error log file on the server.

Precondition: The system is running and the dashboard is working.

Postcondition (success): The errors are logged and displayed in the dashboard.

Postcondition (fail): The system will try to log the error in the error log file and display when the system and dashboard is working again.

Triggering Events: An error occurs somewhere in the system.

Description: Whenever an error occures it will be logged and displayed inside the dashboard *Error logs* section. This section of the dashboard will only be accessible by an admin. The errors are also logged in a file on the server.

FR-12 Cafeteria widget

Goal: Display todays cafeteria menu on the smart TV.

Precondition: A way to access the cafeteria menu has been configured (a pre made configuration for the "KIT Mensa am Adenauerring" will be included) and the cafeteria operates today.

Postcondition (success): The list of dishes will be displayed on the smart TV.

Postcondition (fail): The cafeteria doesn't operate today or the menu could not be retrieved. The menu won't be displayed, instead an error message is shown.

Triggering Events: An admin activates the *Cafeteria widget* and *Cafeteria data plugin* through the dashboard.

Description: The cafeteria menu is retrieved from an external API or website, is parsed and then shown on the smart TV. The widget lists the name of the dish, the price and wether it is vegan / vegetarian / gluten free if such information can be acquired. Additionally the line where the dish can be bought is shown if the cafeteria has such a system.

5.1.2 Optional Requirements

FRO-1 Weather widget

Goal: The widget will display current weather information.

Precondition: Weather data can be retrieved and the plugin is configured correctly.

Postcondition (success): Weather data will be displayed.

Postcondition (fail): An appropriate error message is generated.

Triggering Events: An admin activates the *Weather widget* and *Weather data plugin* through the dashboard.

Description: The widget will display the current temperature and weather type (e.g. "cloudy", "raining"). If the widget size allows it more information is displayed like a rain/snow forecast. Current weather warnings may be shown if such information is available. For this widget to work a location has to be specified in the general config file (FR-6) in a correct format. Additionally an API key for a weather API may potentially be required as well. Multiple locations can be configured and will be cycled through if present. The name of the location is added if multiple locations are to be displayed. No default location exists, if no location is set the widget will display an error message.

FRO-2 Room usage widget

Goal: Deliver information about the availability of rooms. The data will be retrieved from calendars or using other specified data sources (such as websites through web scraping).

Precondition: The rooms and data sources must be configured in order for room usage to show up correctly.

Postcondition (success): The room usage widget shows which rooms are currently in use and which aren't.

Postcondition (fail): The smart TV cannot display the information and shows an error message roughly detailing what went wrong instead. An additional, more detailed, error message is generated which can be seen in the dashboard.

Triggering Events: An admin activates the *Room usage widget* through the dashboard.

Description: Display which rooms are in use and which aren't based on data gathered from calendar events or other data sources.

FRO-3 Printer queue widget

Goal: Show the current entries of printer queues.

Precondition: There are printers on the network which have been properly entered into the *printer queue data plugin* config file.

Postcondition (success): The current printer queue is displayed as list and update regularly.

Postcondition (fail): If the configured printers could not be found instead of the list of print jobs an appropriate error message is displayed and generated for the backend error log (FR-11).

Triggering Events: An admin activates the *Print queue widget* through the dashboard.

Description:

The admin has to register printers manually, which then will be shown in the list. The following is shown for each list item:

- The printer name
- The job ID
- The sender of the print request

FRO-4 Tram / bus schedule widget

Goal: Displays the tram and bus schedule of nearby stations.

Precondition: A number of stations is configured.

Postcondition (success): A list of arriving trams and busses is displayed.

Postcondition (fail): If the data retrieval failed instead of the list an appropriate error message is displayed and logged (FR-11).

Triggering Events: An admin activates the *Tram / bus schedule widget* and the *tram / bus schedule data plugin* through the dashboard.

Description:

For each configured station the data will be fetched automatically by the backend plugin and a list of arriving vehicles is displayed. The following is shown for each list item:

- The type of transportation
- The name of the itinerary
- The starting and destination station
- The time of departure and arrival
- A possible delay

The widget can theoretically also be used for other kinds of transportation if such data can be retrieved. Since gathering data on public transportation can be hard depending on the kind of transportation this functionality is not included out-of-the-box. In this case a new icon for the type of transportation would also need to be added for it to show up correctly in the widget.

FRO-5 Display time of last update

Goal: Display when certain widgets have last been updated.

Precondition: Data has been updated at least once.

Postcondition (success): A little note about when the last update occured is shown for each widget applicable.

Postcondition (fail): Nothing is shown.

Triggering Events: An admin has activated the feature through the dashboard by uploading a config file with the appropriate setting activated.

Description: Add a little indicator to widgets which displays when the data has last been refreshed. The indicator will show the relative time (e.g. “last updated 5 minutes ago”) of when the data was last refreshed. If the feature is active each widget can decide if time of last update should be shown or not and has to do the displaying on its own.

FRO-6 Themeability

Goal: The look and theme of the smart TV GUI is changable.

Precondition: A theme is created and it is set as the active theme in the configuration.

Postcondition (success): The theme is applied

Postcondition (fail): Something is wrong with the theme and the default theme is therefore used instead and the occurance is logged (FR-11).

Triggering Events:

- The active theme configuration was changed.
- An automated theme change occured

Description: Make the system easy to theme. A theme is represented by a CSS file which houses css variables which will be used by the frontend for styling. One theme is active at a time. The current theme is selected by a setting in the general config file (FR-6). Automatic theme changes at specific times can be configured.

FRO-7 RSS feed widget

Goal: Allow the smart TV automatically access RSS feeds of websites

Precondition: A list of RSS feeds has been added by an admin.

Postcondition (success): The content of the RSS feeds are display in chronological order and updated after a configurable amount of time.

Postcondition (fail): If the received data is not valid XML it is discarded. If any other error occurs an error notice is displayed instead and logged (FR-11).

Triggering Events: An admin activates the *RSS feed widget* through the dashboard.

Description: The XML content of a correctly configured RSS feed is displayed. The different RSS feeds are all merged in a chronologically ordered list.

FRO-8 Image display widget

Goal: Be able to display one or more images inside a slot.

Precondition: A list of images has been specfied and all the images can be retrieved.

Postcondition (success): The images are displayed and cycled through with a set time for each image.

Postcondition (fail): If an image fails to be displayed for some reason a placeholder image is displayed which shows an error detailing that an image could not be shown.

Triggering Events: An admin activates the *Image display widget* through the dashboard.

Description: A widget that can display a list of images (called an album). Switching to the next image is done after a preconfigured time which defaults to 15 seconds. This time can only be specified for the whole album, not for individual images.

FRO-9 Server usage widget

Goal: Show the current server usage and workload.

Precondition: Node exporter is configured and up and running.

Postcondition (success): The current usage and workload of different servers is displayed.

Postcondition (fail): If the data retrieval failed instead of the server usage an appropriate error message is displayed and logged (FR-11).

Triggering Events: An admin activates the *Server usage widget* and *Server usage data plugin* through the dashboard.

Description: The current CPU utilization, RAM usage or network throughput of one or more registered servers is displayed in a list. This widget comes in different variants which display different information.

FRO-10 Editing config files in the dashboard

Goal: Be able to edit config files right in the dashboard without having to download and then upload the config file.

Postcondition (success): An admin can edit config files within the dashboard.

Postcondition (fail): An error is logged using FR-11.

Triggering Events: An admin clicks the "Edit"-button in the dashboard.

Description: An admin is able to edit a config file in the dashboard. This is done using a built-in code editor in the dashboard which can be opened for every file that can be uploaded and downloaded. Using the editor is equivalent to downloading the config file and then uploading it again as far as the backend is concerned.

5.2 Non-Functional Requirements

NF-1 Extensibility

The system is extensible and modular.

NF-2 Reliability

The system works without failure if all of the configuration is done correctly. If the configuration is done improperly appropriate error messages are generated but the system stays operational in a potentially limited way by disabling the problematic widgets and/or plugins.

NF-3 Security Requirements

The secure storage of credentials and all other user data is handled by the IDP, which in turn has high security standards as its requirements.

NF-4 Easy to set up and use

The system is usable by people with little technical experience. This includes "good" default configuration files as well as generally easy to understand GUI.

NF-5 Privacy Requirements

HTTPS is used everywhere applicable. Other traffic is encrypted using an appropriate protocol.

6 Global Test Cases

6.1 Global test cases for functional requirements

6.1.1 For the Administrator

GTC-1 Tests FR-1; An admin configures a layout for the smart TV and applies it.

State: An admin is logged in and the *core configuration* view is displayed.

Action: Upload a config file via the *upload* button beneath the label *Layout*. The file should contain a new Layout. E.g. the new Layout could consist of a vertical split and the upper slot is then further split horizontally.

Reaction: The new configured layout is shown on the smart TV.

GTC-2 Tests FR-2; An admin adds a plugin and activates it.

State: An admin is logged in and the *core configuration* view is displayed.

Action: Upload a config file via the *upload* button beneath the label *Plugins*. The file should contain a newly activated plugin. E.g. if not activated before the *Cafeteria widget* (FR-12) could be activated.

Action: Upload a config file via the *upload* button beneath the label *Layout*. The file should contain a layout that contains the newly activated widget. E.g. if the *Cafeteria widget* (FR-12) was activated it could display a single slot with this widget.

Reaction: The activated widget is displayed on the smart TV.

GTC-3 Tests FR-3; An admin changes the system language.

Precondition: A valid language file is in the *i18n* directory.

State: The server is up and running.

Action: Log in and navigate to the *core configuration* via the side bar.

Action: Upload a config file via the *upload* button beneath the label *General*. In the file the language should be set to a new language.

Reaction: The content on the smart TV is displayed in the configured language.

GTC-4 Tests FR-4; An admin configures a widget to only show up at a specified time.

State: An admin is logged in and the *core configuration* view is displayed.

Action: Upload a config file via the *upload* button beneath the label *Layout*. The file should contain a layout that contains a widget for which a display time is specified. E.g. the *Cafeteria widget* (FR-12) is displayed from 10am o'clock to 1pm o'clock.

Reaction: The configured widget is only displayed at the specified time.

GTC-5 Tests FR-7; An admin adds a new calendar to the *Calendar widget*.

Precondition: According to test GTC-2 the *Calendar widget* (FR-7) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Calendar widget* (FR-7) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Calendar data Plugin*. The file should contain a calendar reference.

Action: Upload a config file via the *upload* button beneath the label *Calendar widget*. In the file the newly added calendar is set as active.

Reaction: The *Calendar widget* (FR-7) shows the newly added calendar.

GTC-6 Tests FR-8; An admin adds a website to track for publications.

Precondition: According to test GTC-2 the *Publications widget* (FR-8) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Publications widget* (FR-8) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Publications data Plugin*. The file should contain a website from which the data can be fetched. E.g. the home page of Chair of Embedded Systems.

Action: Upload a config file via the *upload* button beneath the label *Publications widget*. In the file the newly added publications source is set to active.

Reaction: The *Publications widget* (FR-8) shows the newly added publications.

GTC-7 Tests FR-10; An admin configures a timezone for the *Clock widget*.

Precondition: According to test GTC-2 the *Clock widget* (FR-10) has been added and activated.

State: An admin is logged in and the *Core configuration* view is displayed. The *clock widget* (FR-10) is shown on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *General*. In the file a different timezone should be set. E.g. the timezone is set to UTC.

Reaction: The *Clock widget* (FR-10) displays now the new time, accoring to the new timezone.

GTC-8 Tests FR-11; An error happens in the system and is logged in the dashboard.

Precondition: GTC-5 is done, but the server providing the calendar is unreachable.

State: An admin is logged in to the dashboard.

Action: Open the *Error logs* page.

State: The *Error logs* screen is displayed.

Reaction: In the list of errors is an entry saying that the calendar could not be retrieved.

GTC-9 Tests FR-12; An admin configures the *Cafeteria widget*

Precondition: According to test GTC-2 the *Cafeteria widget* (FR-12) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Cafeteria widget* (FR-12) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Cafeteria data plugin*. The file should contain a website from which the data can be fetched. E.g. the website of studierendenwerk for mensa.

Action: Upload a config file via the *upload* button beneath the label *Cafeteria widget*. In the file the newly added Cafeteria menu source is set to active.

Reaction: The *Cafeteria widget* (FR-12) shows the newly added cafeteria menu.

6.1.2 For the Editor

GTC-10 Tests FR-5 and FR-6; A non-administrative user tries to update a configuration option.

Precondition: A user without administrative rights exists.

State: The user is not logged in.

Action: The user logs in and tries to access admin-only APIs using his JWT.

Reaction: The backend rejects the request.

GTC-11 Tests FR-5 and FR-6; An editor tries to log in using the IDP.

Precondition: A user with the role editor exists.

State: The user is not logged in.

Action: The user logs in using the IDP.

State: The IDP sent the user his JWT.

Action: The user authenticates himself to the dashboard using his JWT.

Reaction: The dashboard verifies the JWT and lets the user in.

GTC-12 Tests FR-7; An editor adds a calendar event and waits for it to show up.

Precondition: According to test GTC-2 the *Calendar widget* (FR-7) has been added and activated.

State: An editor is logged in to the dashboard. The *Calendar widget* (FR-7) is displayed on the smart TV.

Action: Open the *Calendar* page.

State: The *Calendar events* screen is displayed.

Action: Enter the title of the event in the *name* field and the location in *location* field. Additionlay, enter the begin time in *from* field and the end time in *to* field. Activate the *Add* button.

Reaction: The newly added calendar event is displayed on the *calendar widget* (FR-7) as soon as it is no more than one day away.

GTC-13 Tests FR-7; An editor deletes a manually created calendar event.

Precondition: According to test GTC-2 the *Calendar widget* (FR-7) has been added and activated. At least one calendar event was added via the dashboard.

State: An editor is logged in to the dashboard.

Action: Open the *Calendar* page.

State: The *Calendar events* screen is displayed.

Action: In the list at the bottom of the screen Activate the *delete* button on the same line as the calendar event you want to delete.

Reaction: The deleted event is removed from the *Calendar events* area and won't be shown in any widget.

GTC-14 Tests FR-9; An editor creates an announcement.

Precondition: According to test GTC-2 the *Announcement widget* (FR-9) has been added and activated.

State: An editor is logged in to the dashboard.

Action: Open the *Announcements* page.

State: The *announcement* screen is displayed.

Action: Enter the message in the *Announcement text* field. Enter the expire time in the *expires in* field, the start time in the *starts in* field and selects the importance in the *Urgency* field. Activate the *save* button.

Reaction: The Announcement is displayed at the set time, for the set duration as described in FR-9.

GTC-15 Tests FR-9; An editor deletes an announcement.

Precondition: According to test GTC-2 the *Announcement widget* (FR-9) has been added and activated. At least one announcement is displayed in the *Past & active announcements* table.

State: An editor is logged in to the dashboard.

Action: Open the *Announcements* page.

State: The *announcement* screen is displayed.

Action: In the list at the bottom of the screen click the *delete* button on the same line as the announcement you want to delete.

Reaction: The Announcement is removed from the *Past & active announcements* area and won't be shown in any widget.

6.2 Global test cases for optional requirements

6.2.1 For the administrator

GTCO-1 Tests FRO-1; An admin sets up a location and API key for the *Weather widget.*

Precondition: According to test GTC-2 the *Weather widget* (FRO-1) has been added and activated.

State: An admin is logged in and the *Plugin configuration* page is displayed. The *Weather widget* (FRO-1) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Weather data Plugin*. The file should contain a location and an API key from which the weather information, for the specified location can be fetched.

Action: Upload a config file via the *upload* button beneath the label *Weather widget*. In the file the newly added weather data source is set to active.

Reaction: The *Weather widget* (FRO-1) shows the current weather for the specified location.

GTCO-2 Tests FRO-2; An admin sets up a calendar for the *Room usage widget.*

Precondition: According to test GTC-2 the *Room usage widget* (FRO-2) has been added and activated. A calendar has been set up in the *Calendar data plugin* (FR-7).

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Room usage widget* (FRO-2) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Room usage widget*. In the file the a room is linked to a calendar from the *Calendar data plugin*.

Reaction: The *Room usage widget* (FRO-2) shows the current availability for every configured room.

GTCO-3 Tests FRO-3; An admin registers a printer for the *Printer queue widget.*

Precondition: According to test GTC-2 the *Printer queue widget* (FRO-3) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Printer queue widget* (FRO-3) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Printer queue data Plugin*. In the file at least one printer should be registered.

Action: Upload a config file via the *upload* button beneath the label *Printer queue widget*. In the file the newly added printer data source is set to active.

Reaction: The *Printer queue widget* (FRO-3) shows the current printer queue of the registered printers.

GTCO-4 Tests FRO-4; An admin sets a station and API key for for the *Tram / bus schedule widget.*

Precondition: According to test GTC-2 the *Tram / bus schedule widget* (FRO-4) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Tram / bus schedule widget* (FRO-4) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *Tram / bus schedule data Plugin*. The file should contain at least one station and an API key to retrieve the public transport information.

Action: Upload a config file via the *upload* button beneath the label *Tram / bus schedule widget*. In the file the newly added stations are set to active.

Reaction: The *Tram / bus schedule widget* (FRO-4) shows a list of the arriving trams and busses for every specified station.

GTCO-5 Tests FRO-5; An admin activates the *Display time of last update* feature.

State: An admin is logged in and the *Core configuration* view is displayed.

Action: Upload a config file via the *upload* button beneath the label *General*. In the file the property *Display time of last update* is set to active.

Reaction: Every widget that has this feature enabled shown not the time since its last update.

GTCO-6 Tests FRO-6; An admin creates a theme and sets it as the active theme.

Precondition: A valid theme CSS file is in the *theme* directory.

State: An admin is logged in and the *Core configuration* view is displayed.

Action: Upload a config file via the *upload* button beneath the label *General*. The *theme* property is set to the new theme.

Reaction: The newly configured *Theme* (FRO-6) is automatically applied.

GTCO-7 Tests FRO-7; An admin adds a list of RSS feeds for the *RSS feed widget.*

Precondition: According to test GTC-2 the *RSS feed widget* (FRO-7) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *RSS feed widget* (FRO-7) is displayed on the smart TV.

Action: Upload a config file via the *upload* button beneath the label *RSS feed data plugin*. The file should contain a list of valid RSS feeds. E.g. the RSS feed RSS für KIT-Fakultät für Informatik.

Action: Upload a config file via the *upload* button beneath the label *RSS feed widget*. In the file the newly added RSS feeds are set as active.

Reaction: The *RSS feed widget* (FRO-7) shows the contents of the RSS feeds chronologically.

GTCO-8 Tests FRO-8; An admin selects an album to be displayed by an *Image display widget.*

Precondition: According to test GTC-2 the *Image display widget* (FRO-8) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Image display widget* (FRO-8) is shown on the smart TV. An Album is already created through the dashboard.

Action: Upload a config file via the *upload* button beneath the label *Image display widget*. In the file the Album property is set to a new Album.

Reaction: The *Image display widget* (FRO-8) displays now the specified Album.

GTCO-9 Tests FRO-9; An admin configures multiple servers for use with the *server usage widget.*

Precondition: According to test GTC-2 the *Server usage widget* (FRO-9) has been added and activated.

State: An admin is logged in and the *Plugin configuration* view is displayed. The *Server usage widget* (FRO-9) is shown on the smart TV. On one or more server Node exporter is up and running.

Action: Upload a config file via the *upload* button beneath the label *Server usage data Plugin*. The file contains all the neccesary configuration to connect and receive information from the target server using Node exporter.

Action: Upload a config file via the *upload* button beneath the label *Server usage widget*. In the file the newly added server are set to be displayed.

Reaction: The *Server usage widget* (FRO-9) displays now the usage of the newly added server.

GTCO-10 Tests FRO-10; An admin edits a config file using the editor.

State: An admin is logged in and the *Core configuration* view is displayed.

Action: Activate the *Edit* button beneath the label *General*.

State: An editor has opened, showing the current general config file.

Action: Change a property, e.g. set the timezone to GTC.

Action: Activate the *Save* button.

State: The editor has closed.

Reaction: The changed property has taken effect as if it would have been changed by uploading a config file.

6.2.2 For the Editor

GTCO-11 Tests FRO-8; An editor creates an album and adds an image to display using the *Image display widget*.

Precondition: According to test GTC-2 the *Image display widget* (FRO-8) has been added and activated.

State: An editor is logged in to the dashboard.

Action: Open the *Images* page.

State: The *Images* screen is displayed.

Action: Activate the *create new album* button.

State: A new album is displayed on the screen.

Action: Enter the name of the new album in the *Album name* field. Activate the *Add* button.

State: The newly created album is shown in the *album list*.

Action: Enter an URL to an image in to the text box beneath the *Add image* label. Activate the *Add* button.

Reaction: The added image is displayed in the album and a thumbnail is displayed.

GTCO-12 Tests FRO-8; An editor deletes an album.

Precondition: According to test GTC-2 the *Image display widget* (FRO-8) has been added and activated. At least one album exists.

State: An editor is logged in to the dashboard.

Action: Open the *Images* page.

State: The *Images* screen is displayed.

Action: Activate the *delete* button at the top right corner of the album you want to delete.

Reaction: The album as well as the images inside are removed from the *Album list*.

GTCO-13 Tests FRO-8; An editor deletes an image from an album.

Precondition: According to test GTC-2 the *Image display widget* (FRO-8) has been added and activated. A list of images has already been added into one album.

State: An editor is logged in to the dashboard.

Action: Open the *Images* page.

State: The *Images* screen is displayed.

Action: Hover over the thumbnail of the images you want to delete and click the x at the top right corner.

Reaction: The deleted images are removed from the album.

7 GUI Design

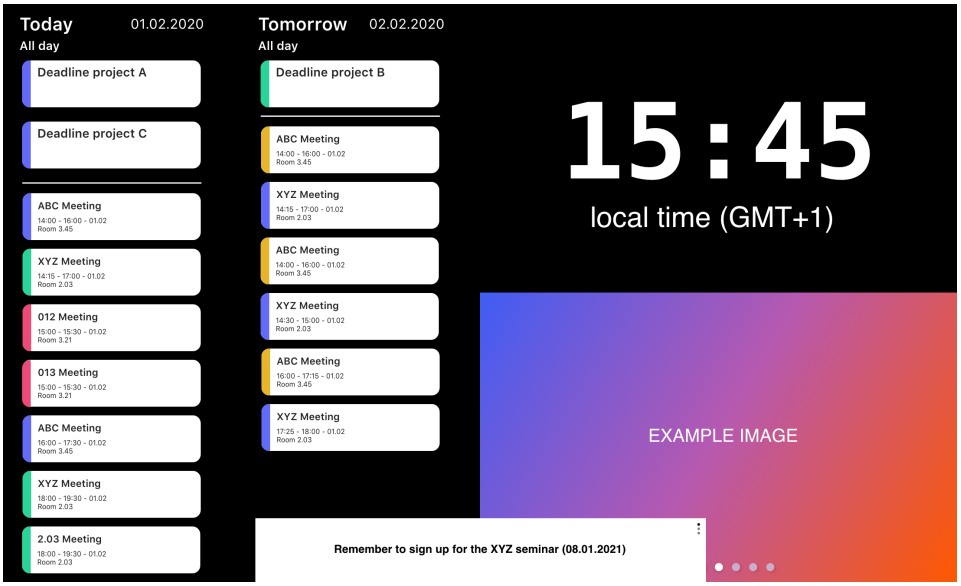
All images are examples of how the implementation will approximately look. They aren't set in stone and don't represent specific features. They show the general idea of how the UI will look like.

7.1 Examples

7.1.1 Calendar, Clock, Image display and Annoucement

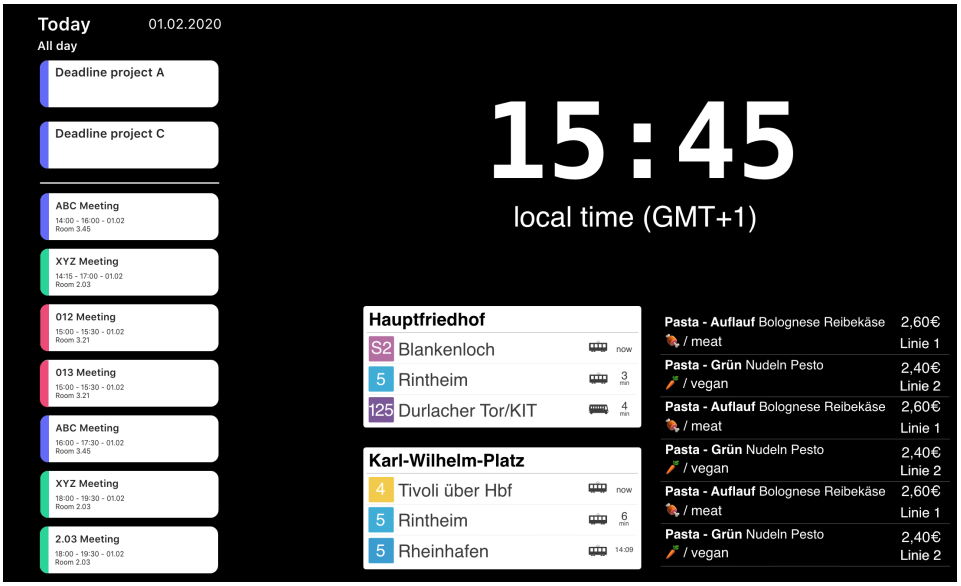
The first example shows the *Calendar widget* (FR-7) on the left and the *Clock* (FR-10) and *Image display widget* (FRO-8) on the right. The right side is split into two slots, the upper one is taken the *Clock widget*, the lower one by the *Image display* widget.

Additionally an *announcement* (FR-9) is displayed on top of the widgets.



7.1.2 Calendar, Clock, Cafeteria and Tram / Bus

The second example shows the *Calendar* (FR-7) and *Clock widgets* (FR-10) once again. This time the space taken up by the *Calendar* is reduced and the *Image display widget* is replaced with the *Tram / bus widget* (FRO-4) and the *Cafeteria widget* (FR-12).

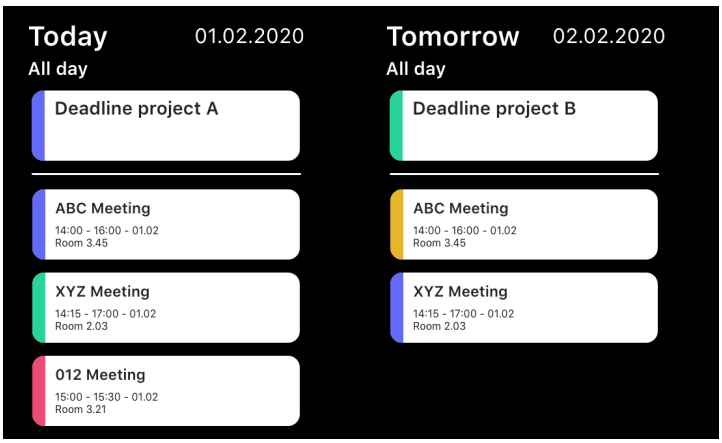


7.2 The Widgets

7.2.1 Mandatory widget designs

FR-7 Calendar widget

The widget shows the next entries into the connected calendar . The entries marked as "all day" are shown separately. The color at the side of the entries indicates the calendar in which they are included.



FR-8 Publications widget

The widget shows the title of the publication, the author(s), publication date and the place where it was published if it is available for each publication.

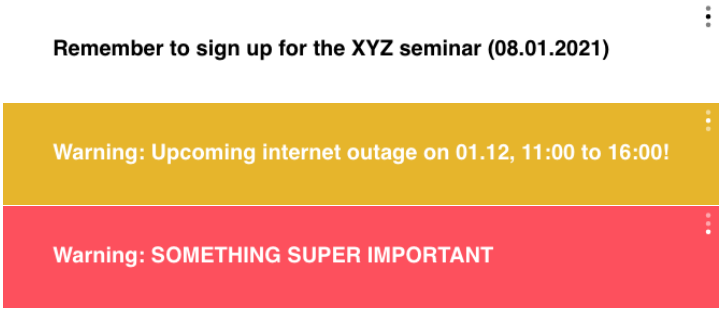
FR-9 Announcements

Announcements are not technically widgets. They cannot be placed into any slots as they are overlayed on top other content or just shrink portion of the smart TV screen taken up by widgets.

The background color depends on the urgency of the announcement and the text color in turn on the background color. The text color is either black or white, depending on

what gives the greater contrast (this is not something that is being actively computed; this is something the theme author needs to manually check and build in).

The dots in the top right corner indicate which announcement is being shown and how many announcements are currently active. If only a single announcement exists the dots are hidden. The color of the dots may also depend on the background color.



FR-10 Clock widget

The clock widget can show either 12h or 24h time. Which is displayed depends on the language settings in the "General"-config file or the property passed to the instance of the widget.

FR-12 Cafeteria widget

The name of the meal, the price and where it can be bought is shown. If additional information about the meal is present it is shown (e.g. vegan, vegetarian, gluten free).

Pasta - Auflauf Bolognese Reibekäse 🍖 / meat	2,60€ Linie 1
Pasta - Grün Nudeln Pesto 🥕 / vegan	2,40€ Linie 2
Pasta - Auflauf Bolognese Reibekäse 🍖 / meat	2,60€ Linie 1
Pasta - Grün Nudeln Pesto 🥕 / vegan	2,40€ Linie 2
Pasta - Auflauf Bolognese Reibekäse 🍖 / meat	2,60€ Linie 1
Pasta - Grün Nudeln Pesto 🥕 / vegan	2,40€ Linie 2

7.2.2 Optional widget designs

FRO-1 Weather widget

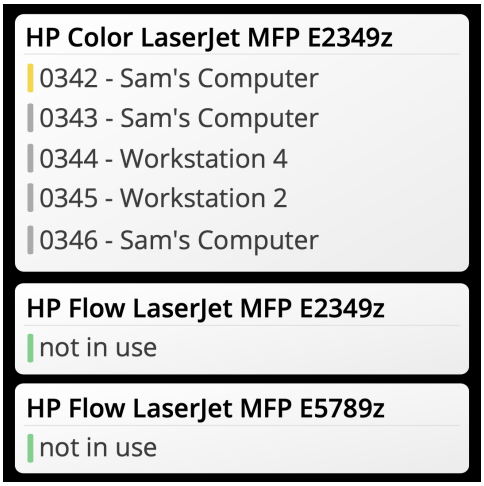
The widget shows the current temperature and weather type (like snow, rain,...) as well as the current location. If present like in this example weather warnings are shown too.

FRO-2 Room usage widget

For each room the name is shown as well as its current status. The status is either “free” or “taken”.

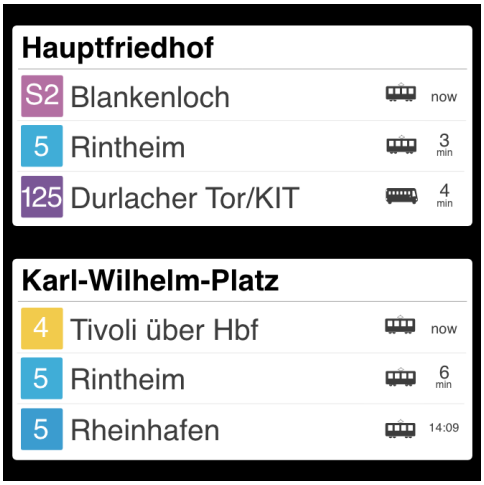
FRO-3 Printer queue widget

The widget shows the current job lists of the registers printers with their name. For every job list the job IDs are displayed together with the name of the sender of the print request.



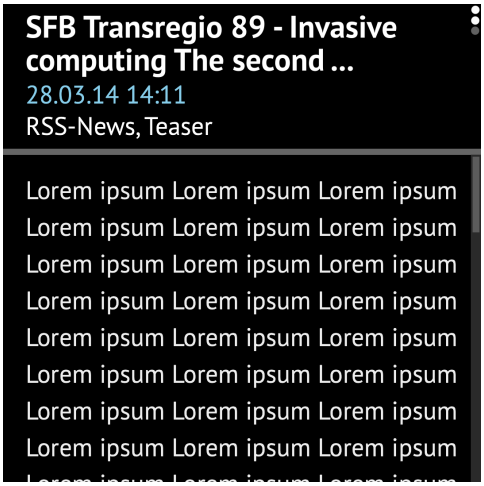
FRO-4 Tram / bus schedule widget

For each station the connecting trams and busses are shown. Their time of arrival, name and direction are shown. The icon indicates wether it is a tram or a bus.



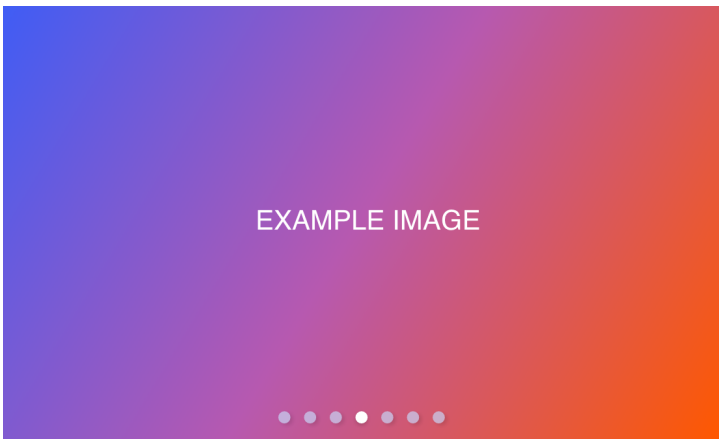
FRO-7 RSS feed widget

Every RSS feed will show the entries of the feed.



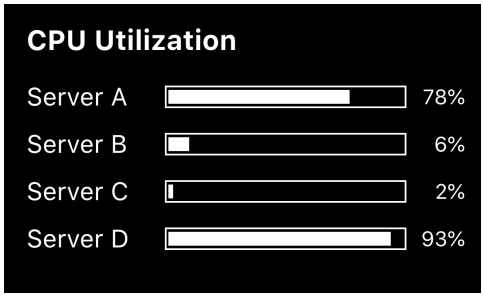
FRO-8 Image display widget

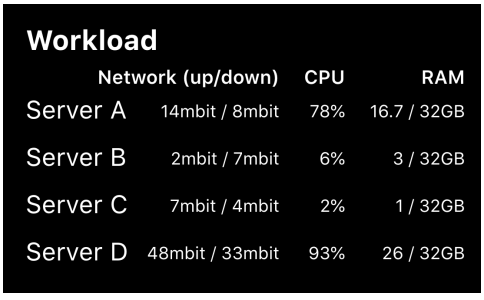
The dots on the bottom indicate which photo is currently being shown.



FRO-9 Server usage widget

The widget comes in 3 variants. The first one shows the CPU utilization across all tracked servers, the second one the RAM usage and the third one multiple statistics at once. It displays network throughput, CPU utilization and RAM usage.





7.3 The dashboard

The dashboard consists of multiple pages, each of them accessible via the sidebar (if the logged in user is an admin, for editors only the last 3 are accesible):

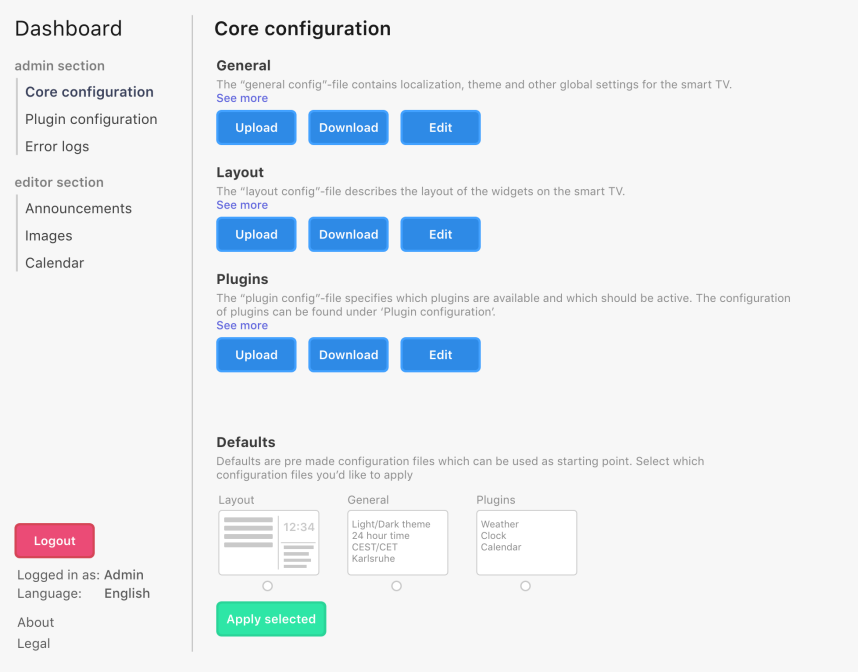
- Core configuration
- Plugin configuration
- Error logs
- Announcements
- Images
- Calendar

7.3.1 Mandatory dashboard designs

Core configuration

The "Core configuration"-page allows uploading and downloading of the General-, Layout- and Plugins config file. The page also allows applying premade config files. A link to the relevant documentation about the specific config file is provided.

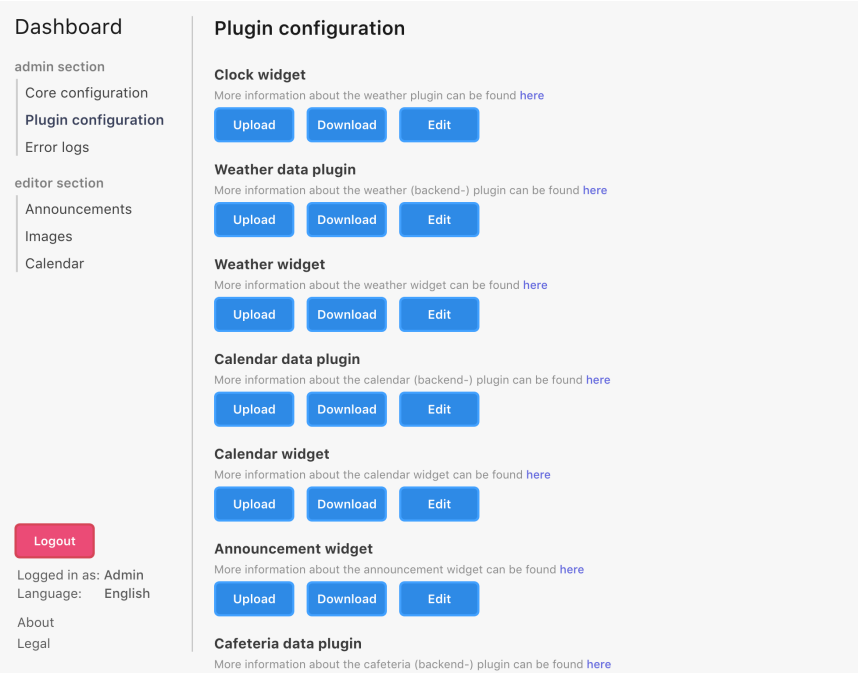
If FRO-10 is implemented editing the files in place is also available using the *Edit*-button, otherwise this button is not shown.



Plugin configuration

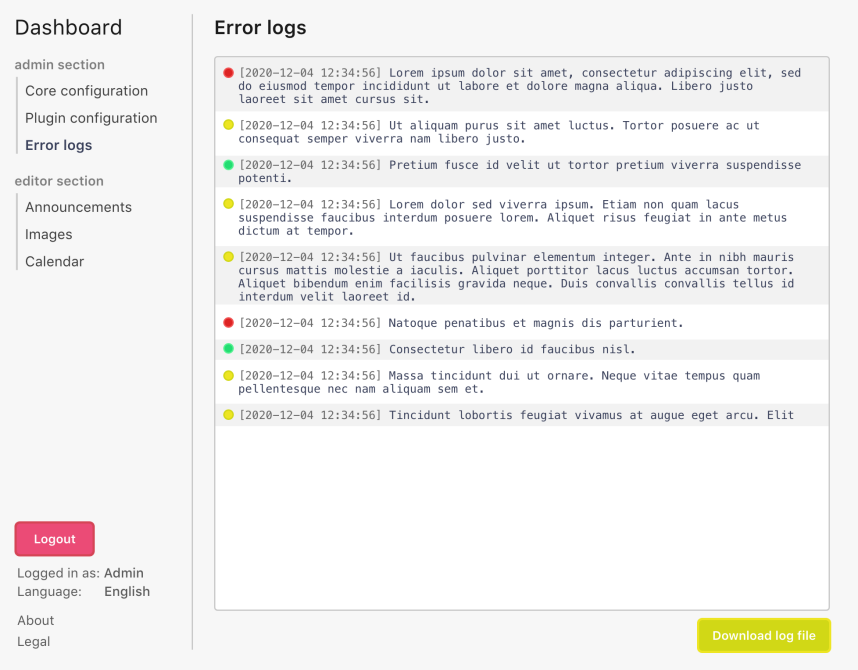
The “Plugin configuration”-page allows uploading and downloading config files for each plugins.

If FRO-10 is implemented editing the files in place is also available using the *Edit*-button, otherwise this button is not shown.



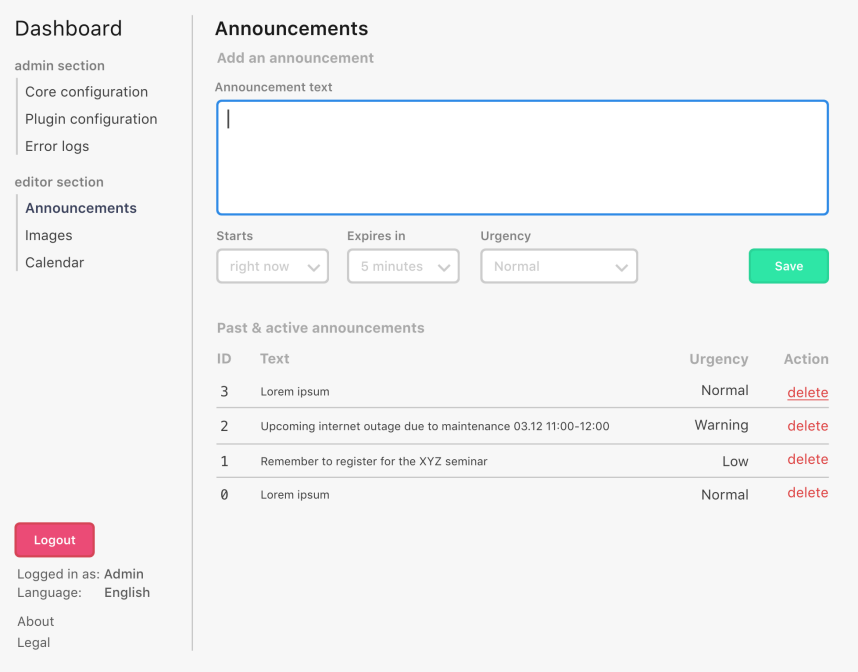
Error logs

The “Error logs”-page shows generated error messages an their urgency.



Announcements

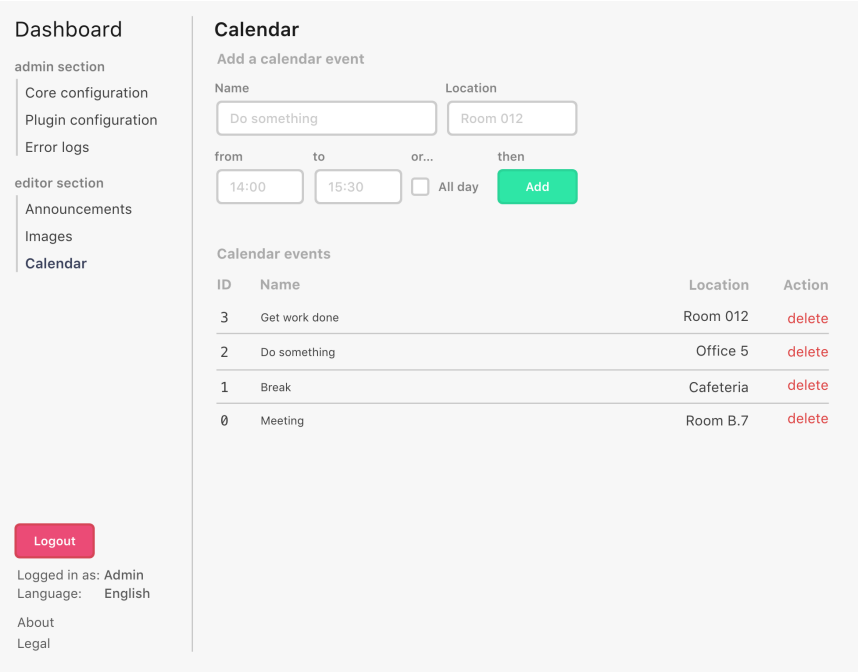
The “Announcements”-page allows admins as well as editors to create new announcements and delete old ones. In the image the input element for the announcement text is focused and the hover effect of the “delete”-button in the table is shown.



Calendar

Calendar events can be manually added using the “Calendar”-page. Existing calendar events which haven’t passed yet can be deleted using the “delete”-button in each table row.

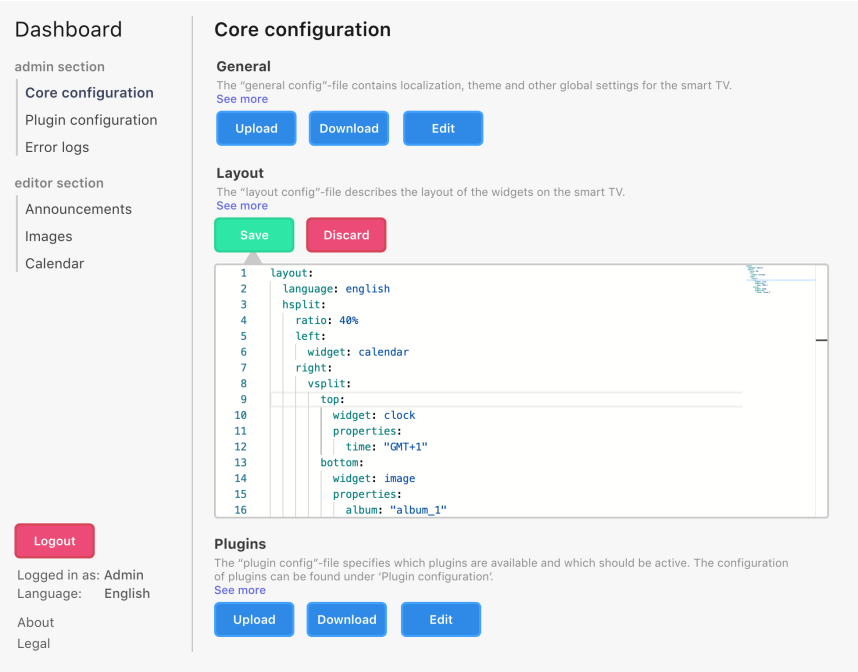
Calendar events can be manually added using the form in the top half of the page. When “All day” is checked both the “from” and “to” input fields are disabled.



7.3.2 Optional dashboard designs

Editing config files

If FRO-10 is implemented editing of the config files in place is possible. To do so click the “Edit”-button, edit the file and then either hit “Save” or “Discard” to save or discard the changes.

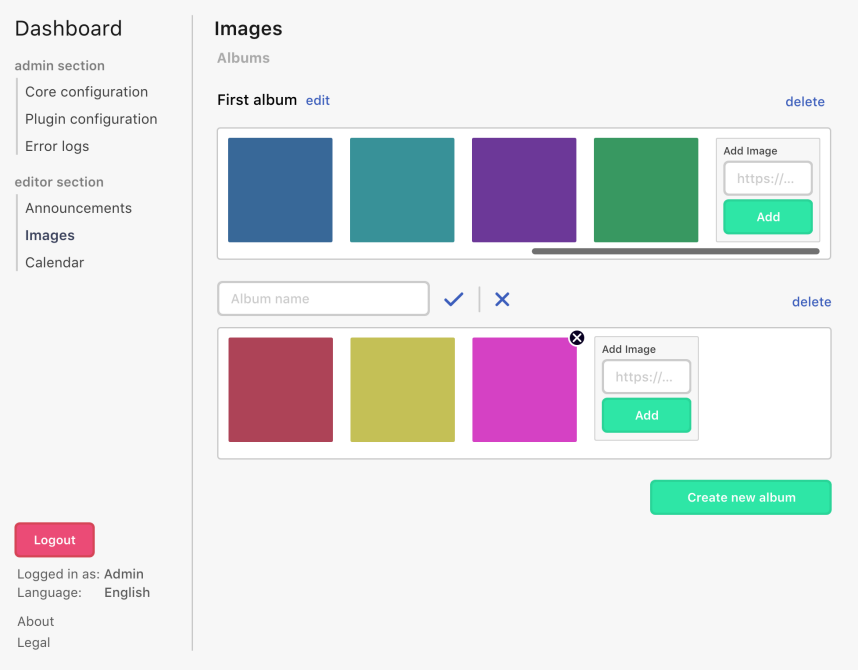


Images

Albums can be created and photos added to albums by URL on this page. A preview of each image is given.

Albums can be deleted using the "delete"-button, they can be renamed using the "edit"-button besides their name. Using the "checkmark"- and "cross"-icons changes to the name can be accepted or cancelled. A new album can be created using the "Create new album"-button.

Individual images can be removed from an album by hovering over them and pressing the "cross"-icon in the top right corner of the image.



Glossary

admin An administrator as determined by the IDP. Generally has the most privileges and is in charge of configuring the system.

announcement A message added by an authorized user in the dashboard to be displayed on the smart TV.

API An "application programming interface" is an interface with other services like getting weather information for a given location.

backend The API which runs on the server and serves the frontend.

calendar A calendar accessible by either iCal or CalDav.

calendar event A calendar event is a specific event added to a calendar. This also includes manually adding an event using the dashboard to the server's internal calendar.

config file A config file is a file which contains properties that describe some sort of setting in a formal and orderly way according to a strict specification. This project relies on multiple config files.

CSS Cascading Style Sheets are the primary method HTML is styled.

dashboard A website used for configuring the frontend and backend. In this product the dashboard is defined through FR-6.

database A sql database. In this system it refers to the database that can be seen in the General Concept diagram.

editor An editor as determined by the IDP. Generally less permissions than an admin but more than a viewer.

frontend The webapp running on the smart TV.

GUI The "graphical user interface" is everything shown to the users in an visual way.

HTTP polling HTTP traffic is inherintly unidirectional; the client sends a request and the server answers. The server never sends anything on its own. HTTP polling is a way to circumvent this. Short polling describes is a scenario in which a client periodically sends requests asking for new data. Long polling has the client sent a request for new data to which the server does not respond directly. It waits until new data is available and only then responds (this is called holding a request). As soon as the client receives a response it sends another request for new data.

IDP An Identity Provider is a service that handles authentication and authorization of users.

JWT JSON Web Tokens are an open, industry standard RFC7519 method for representing claims securely between two parties. See jwt.io for more.

LTR Left-to-Right: See the Unicode Specification section 2.10.

plugin A plugin is a software component that adds a new feature to the system. One example for the goal of a plugin is adding a new widget.

RSS feed An RSS feed is a standardized web feed. It may be used for things updated frequently like blogs and podcasts. It can be seen as a highly customizable decentralized notification system.

RTL Right-to-Left: See the Unicode Specification section 2.10.

server The server as described in this document.

server usage and workload The usage and workload of connected servers. This is not intended to specifically track the server (/ servers) running this system; this is meant as a general tool to track the workload of all kinds of servers.

slot A position on the smart TV that can eventually be filled by a widget.

smart TV The television for which a webapp is described in this document.

startup The process in which the system is being started.

timezone name Abbreviations for timezones based on this list or timezone offsets.

viewer A viewer as determined by the IDP.

widget An individual module that is to be displayed on the smart TV.