Lightweight Information System Technology LIST

Simon Umbricht, Christoph Keller

November 17, 2008

Assisted by:
Prof. Dr. Bernhard Plattner
Dr. Vincent Lenders
Bernhard Distl
Dr. Franck Legendre

Abstract

The main goal of this project was to develop a server-based information system using infoscreens. These infoscreens will be located near the entrances of the ETH buildings of D-ITET and D-MAVT. The main problem with the present solution was that most people would not read informational mail anymore, resulting in a severely obstructed flow of information. The solution chosen to alleviate this problem were the mentioned infoscreens due to their presence, up-to-dateness and manageability.

The design goals were to build a user-friendly, centrally administered system with relatively low hardware requirements compared to commercially available systems. Furthermore, easy integration into pre-existing content management systems was a key requirement to facilitate switching to the new system.

Contents

1 Background / Motivation

Over the last few years, the AMIV[1] experienced an increasing problem in communication. As they provide a lot of events every semester, quite a few E-mails are sent to inform the students about upcoming events. Unfortunately, these E-mails are largely ignored if not filtered by the Spam filter. As a result, a few events have had a very small attendance. To counter this, many possibilities were discussed. The most promising solution was to build an infoscreen system with a centralized content management system to distribute the information and keep the students informed.

After informing the department administration of the D-ITET[2] and D-MAVT[3] about this project, the D-ITET wanted to have an additional screen to distribute their information as well. In return, they offered to pay some of the costs.

Additionally, the D-ITET is looking for a way to make its main building ETZ more attractive by bringing in some life and a few neat technical features that one would expect at the location of a technically focused department. In that sense the infoscreens would be the first element on the way to achieve this goal.

2 System Description

The whole system is based on three parts

The first part is made up by the screens. They are the only part that can actually be seen by the public and they should therefore be as large as possible but also as low power consuming as possible. Furthermore, they must be configurable remotely in an straightforward fashion. Configuration should mainly be possible through a web interface rather than via SSH[4] or direct database access.

The second part is the backend. Realized through a database, all the information and configurations are stored and managed centrally. As to circumvent distributing database passwords to the individual screens, the content for each screen is generated centrally by a PHP script running on a web server. This way, content is delivered to the screens in the form of RSS feeds which make the whole system usable for PodNet[5] and even for other applications such as any RSS reader. Every feed on the screen can also be subscribed to as an RSS feed in another application.
Furthermore, there are some scripts to automatically import content to the database, such as information stored in another database table.
The last part is the frontend. Many scripts are needed to facilitate the configuration. There are some scripts to manage the content of the individual feeds displayed on the infoscreens, some others to define which feed gets displayed when and where. Currently, the division of the screen into different screen

[1]Akademischer Maschinen- und Elektroingenieurverein
[2]Department of Information Technology and Electrical Engineering
[3]Department of Mechanical and Process Engineering
[4]Secure Shell. A network protocol allowing data to be exchanged using a secure channel between two networked devices
[5]Podcasting framework for mobile distribution of user-generated content. http://podnet.ee.ethz.ch

regions has to be done by hand in the database as this is by far not a daily maintenance task.

3 Display Entities

The screens form the main components of this system. They display the data retrieved from the server and they are based on two physical components. The first component is a small mini ITX board which acts as the controlling unit of the screen. The second component is a 22" TFT display.

3.1 Hardware

3.1.1 Controlling Unit

After a hint and a short evaluation, suitable hardware was found. A small mini ITX board with an AMD Geode chip set from PC-Engines[6], alix1c, is now used for the system. Table 1 shows the hardware specifications of alix1c[7].

Table 1: Hardware specifications of the alix1c

CPU	500 MHz AMD Geode LX
DRAM	256 MB SDRAM on board
Storage	4 GB Compact Flash
Power	12V DC, DC-DC converter on board. No bulky ATX PSU needed.
Expansion	miniPCI + 3.3V PCI + LPC + optional I2C
Connectivity	1 Ethernet port (Via VT6105M 10/100)
I/O	2 COM, 4 USB, 1 LPT, audio, VGA
Board size	6.7 x 6.7" (miniITX), low profile
Firmware	Award BIOS

The main advantage of this board are its dimensions and the low power consuming design. The normal power consumption is listed with `0.4 A @ 12 V` with a peak power consumption of `12 W` [1]. Furthermore, one big advantage is the x86 architecture. This allows the use of precompiled x86 software.

3.1.2 Display

The largest possible resolution, the Geode LX chip set can provide, is `1680 x 1050`. One requirement was to have the screen as large as possible. Therefore the 22" wide screen display were suitable. After evaluating different monitors (see section 8.1) the decision was made to use the SAMSUNG SyncMaster 226cw.

Table 2 shows the specifications of the SAMSUNG SyncMaster 226cw [8]

[6]http://www.pcengines.ch
[7]from [1]
[8]from [2]

Table 2: Specifications of the SAMSUNG SyncMaster 226cw

Screen Size	22" Wide
Resolution	`1680 x 1050`
Brightness	`300 cd/m`^{2}
Contrast Ration	DC `3000:1`
Video Signal input	Analog RGB, DVI
Power Consumption	50W (Max)

3.2 Software

3.2.1 Operating system

The operating system used is a specially adapted version of Debian Etch. The adaptations are basically optimizations for the use with a flash card as storage. In particular, directories with fast changing content such as `/tmp` or `/var/log/` are set up as ram disks and the `noatime` flag[9] is set for all mounted file systems. Furthermore, much of the software which is installed by default was removed.

To display the contents in an appealing way, an Xserver and the desktop environment GNOME is used. A configuration without GNOME was tested, but did not work satisfyingly as the Firefox browser did not work in full screen mode as expected.

After startup, the user AMIV is logged in automatically, the standby script (see section 3.2.4) and Firefox are started with a custom extension (see sections 3.2.2 and 3.2.3) as default page.

3.2.2 Firefox

To display the information, Firefox is used together with a specially programmed extension (see section 3.2.3 for a detailed description of the extension). Extensions are simple pieces of code that can be installed by the user to have additional functionality, i.e. an integrated download manager or an FTP client. Using Firefox is an excellent way to keep the programming task manageable; Firefox offers a fully functional environment, in which interfaces are provided for virtually any relevant task. Thus, using these interfaces saves a lot of time compared to programming everything oneself. In order to make full use of these feautures, a simple HTML page together with JavaScript would not suffice, since Firefox blocks access to many interfaces for non-trusted code; to be considered trusted, the code has to run in the context of an extension.

As Firefox' Gecko rendering engine offers good service for HTML/XML based documents, the task of displaying the content was left to Gecko. Formatting the content with HTML and CSS is much easier and less error-prone than developing another rendering algorithm.

Firefox' integrated XML parser was also a great help, as it has the ability to perform all the XML related tasks needed to put together a final XML document containing the whole structure and content to be displayed on the screens. It allows parsing an XML document directly from a specified URL and the interface

[9]prevents setting of access time after a file access

for reading out parts the document's tree structure is very easy to use. Copying certain parts of the tree from one document to another facilitates the assembly of the document to be displayed; subtrees of an RSS feed (corresponding to individual display items on the screen) can be inserted into the XML document using just a few commands.

Furthermore, the extension *Full Fullscreen*[10] is used to start Firefox in full screen mode without any frames and bars visible.

3.2.3 Infoscreen Firefox extension

This custom Firefox extension is responsible for a multitude of different tasks linked to display all the content optimally. The main reason why using a custom extension to render the content was favored over a plain HTML is, that extensions have full access over a broad palette of tools offered by Firefox only to "privileged code". The one tool the extension heavily relies on is Firefox' XML parser, which is used to parse the (XML compliant) RSS feeds as well as a server-generated, XML based configuration file (see below).

In a predefined interval of 1 minute, the extension fetches an XML document from the server (generated by a PHP script) that includes the information, which feeds are to be displayed and in which screen region. Additionally, the document also contains an XML node specifying the whole screen layout, i.e. it contains the whole XUL[11] structure that makes up the division of the screen into screen regions. This is achieved by nested `hbox` and `vbox` tags (container boxes with horizontal/vertical alignment for their child nodes. See Fig. 1 for an example employing boxes); each box serves as a container for other boxes (such as to divide the box) or feed content to be displayed. Each screen region also has a display type associated to it that defines how the content is to be displayed. Currently there are the following different self-defined display types:

- MainScreen: All items contain a title, a description, a box with date, location and a link, and optionally an image to be placed below the title. (Fig. 2)
- CalendarScreen: The Items are ordered by date and grouped by day. Only the date/time and the title of the items are shown. Items that don't contain a valid date in the future are discarded. (Fig. 3)
- TickerScreen: Designated area for a few short messages including title and description. (Fig. 4)

New display types can be created by writing a new class inherited from the class GenericScreen. Thus, a change of screen layout or the list of feeds involved will take at most a minute to propagate from the database to the screens.

The content of all the feeds are parsed and item-wise inserted into the aforementioned XUL structure. In the probable case that there are more items to be displayed that there is space on the screens, some kind of filtering needs to take place to pick items to be displayed while ensuring fairness and avoiding starvation[12]. As many items as possible are selected in a "priorized random"

[10]see https://addons.mozilla.org/en-US/firefox/addon/1568 for details

[11]XML User interface Language, the Mozilla's XML based language to create user interfaces

[12]Starvation is the case where an item never gets displayed because of the presence of higher-priority items

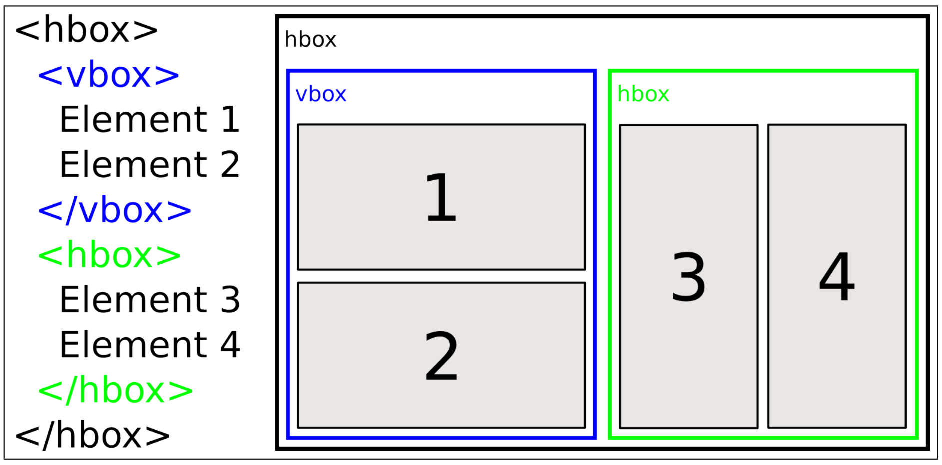
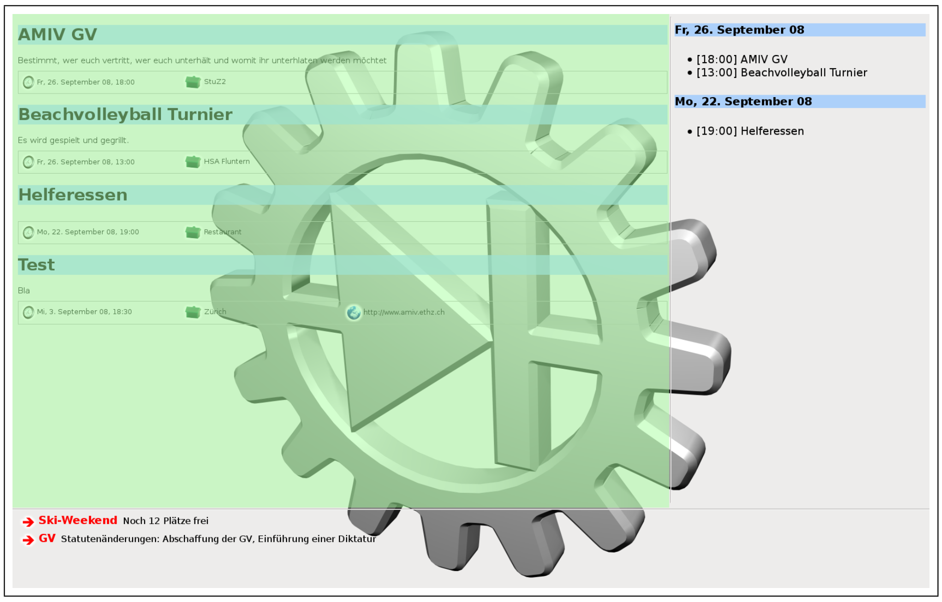


Figure 1: Example employing hbox and vbox



AMIV GV

Bestimmt, wer euch vertritt, wer euch unterhält und womit ihr unterhlaten werden möchtet

Fr, 26. September 08, 18:00 StuZ2

Beachvolleyball Turnier

Es wird gespielt und gegrillt.

Fr, 26. September 08, 13:00 HSA Fluntern

Helferessen

Mo, 22. September 08, 19:00 Restaurant

Test

Bla

Mi, 3. September 08, 18:30 Zürich http://www.amiv.ethz.ch

Fr, 26. September 08

- [18:00] AMIV GV
- [13:00] Beachvolleyball Turnier

Mo, 22. September 08

- [19:00] Helferessen

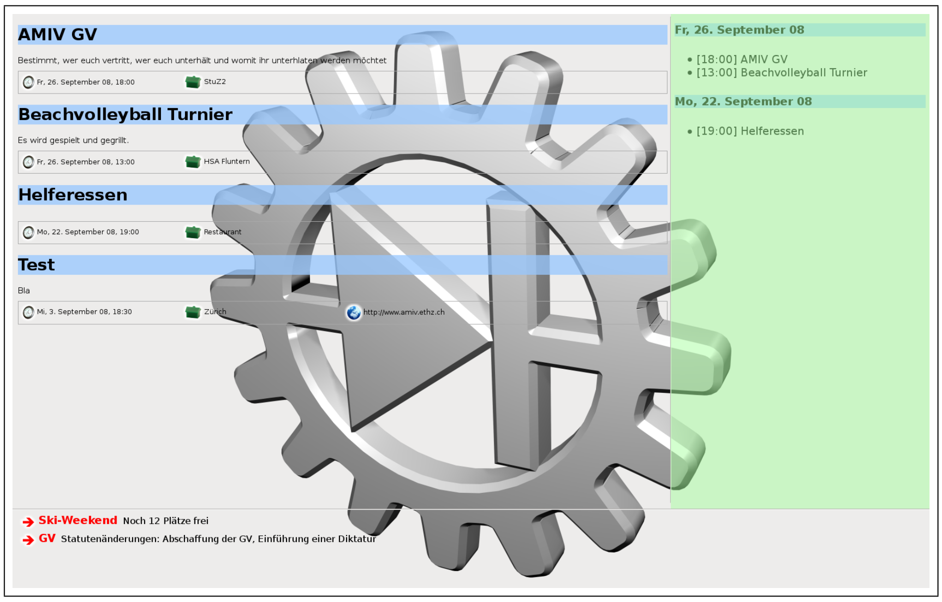
Ski-Weekend Noch 12 Plätze frei

GV Statutenänderungen: Abschaffung der GV, Einführung einer Diktatur

Figure 2: MainScreen

fashion, meaning that items with higher priority have a higher chance of being selected.

The detailed algorithm chosen is best explained using an example: There are four items A, B, C and D with priorities 10, 10, 20 and 5 respectively. Summing up all the priorities results in 45, thus a floating point random number between 0 and 45 is used to select a first item. Item A is selected if the random number is between 0 and 10, B if it's between 10 and 20, and so on. Let us assume that there is only space enough for two items on the screen and C is selected in the first round. Now, a second round takes place with only items A, B and D and


AMIV GV

Bestimmt, wer euch vertritt, wer euch unterhält und womit ihr unterhlaten werden möchtet

Fr, 26. September 08, 18:00 StuZ2

Beachvolleyball Turnier

Es wird gespielt und gegrillt.

Fr, 26. September 08, 13:00 HSA Fluntern

Helferessen

Mo, 22. September 08, 19:00 Restaurant

Test

Bla

Mi, 3. September 08, 18:30 Zürich http://www.amiv.ethz.ch

Fr, 26. September 08

- [18:00] AMIV GV
- [13:00] Beachvolleyball Turnier

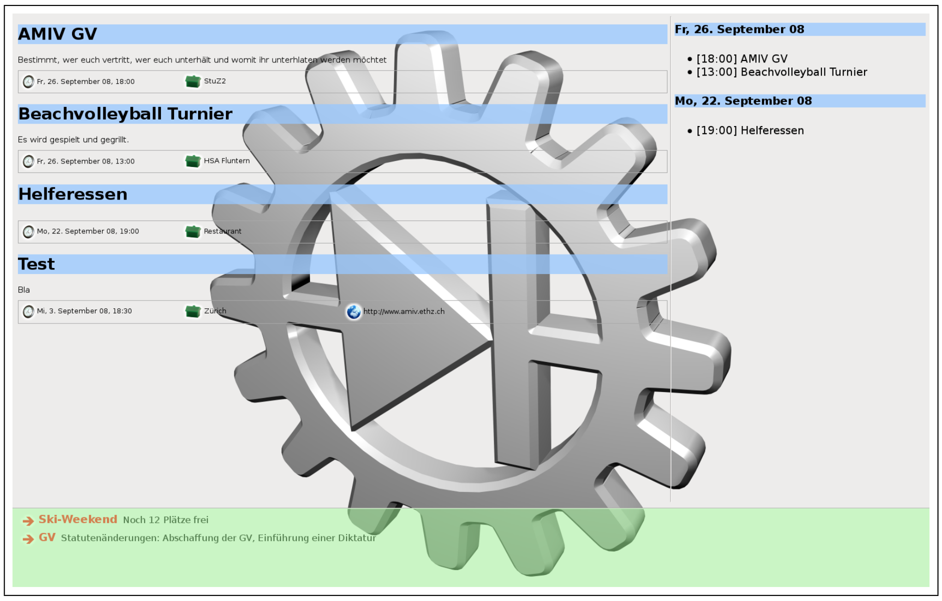
Mo, 22. September 08

- [19:00] Helferessen

Ski-Weekend Noch 12 Plätze frei

GV Statutenänderungen: Abschaffung der GV, Einführung einer Diktatur

Figure 3: CalendarScreen


AMIV GV

Bestimmt, wer euch vertritt, wer euch unterhält und womit ihr unterhlaten werden möchtet

Fr, 26. September 08, 18:00 StuZ2

Beachvolleyball Turnier

Es wird gespielt und gegrillt.

Fr, 26. September 08, 13:00 HSA Fluntern

Helferessen

Mo, 22. September 08, 19:00 Restaurant

Test

Bla

Mi, 3. September 08, 18:30 Zürich http://www.amiv.ethz.ch

Fr, 26. September 08

- [18:00] AMIV GV
- [13:00] Beachvolleyball Turnier

Mo, 22. September 08

- [19:00] Helferessen

Ski-Weekend Noch 12 Plätze frei

GV Statutenänderungen: Abschaffung der GV, Einführung einer Diktatur

Figure 4: TickerScreen

a random number between 0 and 25, according to the scheme described before. This algorithm ends as soon as there are no more items to be displayed or there is no item that could still fit on the screen. The surplus items are removed and will be considered for display again the next time the algorithm is invoced. The frequency of these content updates can be set seperately for each display region.

3.2.4 Display standby

Since it is a waste of power to always run the display 24/7, a small script is running to turn off the display at a configured time and turn it on again at a different time. (See standby.php for more details). The configuration is saved in an MySQL database and can be changed with a config tool (see section 5.2).

3.2.5 Lockdown, Openup

Two scripts are used to secure the boxes from outside attacks and give back control for configuration/debugging purposes. By running `lockdown`, the keyboard and mouse are basically deactivated on the xserver. Furthermore, the cursor is changed to a transparent one and `Ctrl-Alt-Backspace` is deactivated as well. `openup` reverses these changes. As these scripts copy around configuration files, they need to be run as user root. Furthermore, they copy predefined configuration files to the places they are in use, so this has to be taken into account when changing configuration files on the board itself. To open the locked system there are ways. The system can be unlocked either through ssh or one can us one of the console terminals by pressing e.g. `Ctrl-Alt-F1`.

4 Backend

4.1 Data Base Structure

4.1.1 Screen Layouts

The screen layouts are defined and assigned to the screens through the database tables `infoscreen_screenLayouts` and `infoscreen_screens` described in Tab. 4 and Tab. 3 respectively.

Every layout has a numeric ID that can be assigned to a screen using the table `infoscreen_screens`. To facilitate the management of these entries, a regular expression is used to link the layout IDs to the screen names describing location and type of the screen. The AMIV's screen in the foyer of the ETZ building could therefore be named `etz_foyer_amiv` and to use layout 3 for all AMIV screens in ETZ, setting the regular expression `^etz_.*_amiv` for layout 3 would do the job.

As a screen is typically made up of 2 to approximately 5 screen regions, there needs to be a flexible solution that allows a dynamic number of screen regions that is still manageable. Different screen layouts allow separate screens to be split up differntly. For maximum flexibility, a nested box model has been chosen where several entries sharing the same layout ID make up one screen definition. Each entry corresponds to one box in the screen layout (see Fig. 1) as well as one node in the XUL tree structure that is displayed. These nodes serve as a parent node under which more nested boxes or display items can be attached. Nesting is achieved in the following way:

- One root node (with parent node set to NULL) is allowed per screen layout. This represents the whole screen and acts as the container for further subnodes.
- Each node specifies an orientation (horizontal or vertical; `hbox` or `vbox` respectively), telling in which direction its child elements are to be placed.

If a region is to be split horizontally as to accommodate two or more regions side by side, horizontal would be the choice here

- The node ID is a name describing the content or type of the respective screen region. As it is used primarily to build up the screen layout structure it can be chosen freely but must be unique. This ID is also used later on to specify in which screen region a feed is to be displayed (`displayID` in Tab. 6)

As this does not represent a regular maintenance task, there is currently no web frontend to change the screen layouts. With the information provided in this section and database access it should however be easily manageable manually.

Table 3: Screen Layout Assignments (infoscreen_screens)

Field name	**Data type**	**Description**
layoutId	int(5)	ID of the screen layout to be applied to location
locationRegexp	varchar(40)	Regular Expression specifying, which displays to assign the layout to

Table 4: Screen Layout Definitions (infoscreen_screenLayouts)

Field name	**Data type**	**Description**
layoutId	int(5)	Layout ID, to which the screen region belongs. This ID is NOT unique
nodeId	varchar(40)	Name of the region's top level node.
parentNodeId	varchar(40)	Name of parent's node, or NULL for top level nodes (one per layout)
displayType	varchar(30)	Type/class of the display region
refreshTime	int(10)	Time in seconds between two content updates
orient	h/v	Orientation for the placement of child elements
height	int(5)	Height of the region, in pixels
width	int(5)	Width of the region, in pixels
order	int(5)	Ordering of elements between sibling regions with same parent

4.1.2 Feeds

The table `infoscreen_feeds` stores some meta data about those feeds that are locally managed using the tools described in section 5.1.1. This information is used to export an RSS compliant feed from the database content. Ready-made RSS feeds that are imported from some external source (e.g. a list of diploma theses or any other information channel that is provided by someone else) on the other hand have no entry here, as they cannot be managed.

In order to be displayed, every feed needs to be entered in the table `infoscreen_feedDisplay`, which specifies which feed should be displayed when and where. To provide some flexibility, regular expressions are used for the where (see section 4.1.1). The when is determined by a general policy (display or don't display) together with time-dependent exceptions to the policy. All exceptions are kept in the table `infoscreen_displayExceptions` shown in Tab. 7. To illustrate this, consider the mensa menu, which is only of interest during lunch time; the key idea is not to display it, except for Monday–Friday, 11:00–13:00. In this case the policy would be not to display it and for each week day there would be one exception. This would of course also work for feeds that are to be displayed always except for certain times.

Tab. 8 shows the structure of an individual feed item.

Table 5: Feed Definitions for locally managed feeds (infoscreen_feeds)

Field name	**Data type**	**Description**
id	varchar(30)	Feed ID
title	varchar(50)	Feed title
link	varchar(150)	Link to a web site for more information
editor	varchar(100)	Editor responsible for management of the content. Preferably *xxx@yyy.zzz (Name)*

4.2 Mensa Feed

One item that should be displayed around lunch time is the current menu from the mensa. There already exist RSS menu feeds for every mensa. Mainly, there are two different kinds of feeds. One provides a list of every menu within the next two weeks for a specific mensa. Another provides the menu of the day. Unfortunately, in the daily feed, every menu has its own RSS item. This would obstruct the whole screen; having only one item for all the menus would be much more compact. Therefore a new feed generator was needed. It can be seen on
`http://www.amiv.ethz.ch/infoscreen/mensa.php?mensa=...`

The three mensae *Gloriabar*, *Tannenbar* and *Clausiusbar* are configured, but more can be included very easily. The script parses the feed from the official mensa website `http://www.gastro.ethz.ch/meals/rssfeeds`, takes the four main menus and puts them into one RSS item.

Table 6: Feeds to be displayed (infoscreen_feedDisplay)

Field name	**Data type**	**Description**
id	int(5)	Unique ID number purely for manageability reasons
feedURI	varchar(150)	URI describing where the feed can be found
displayID	varchar(40)	ID of the display region, where the feed is to be displayed. This has to be a nodeID from Tab. 4
priority	float	Priority of the feed, used to give higher (global) priority to certain feeds as a whole, whereas (local) priority can be modified on per item basis
displayByDefault	int(1)	Whether or not a feed should be displayed per default. Whenever an exception occurs (see below), this behavior is inverted
displayExceptionsID	int(5)	ID of the set of exceptions that should be considered for this feed
active	int(1)	1 if the feed is active and should be diplayed, 0 otherwise
locationRegexp	varchar(40)	Regular Expression specifying, on which displays to show the feed

Table 7: Display Exceptions (infoscreen_displayExceptions)

Field name	**Data type**	**Description**
id	int(5)	Unique ID number purely for manageability reasons
exceptionID	int(5)	ID of the set of exceptions. This is NOT unique
weekday	int(1)	1=Monday, . . . , 7=Sunday
from	time	Daytime, from when the exception is active
until	time	Daytime, until when the exception is active

5 Frontend

5.1 Data Management Tools

5.1.1 Feed Management

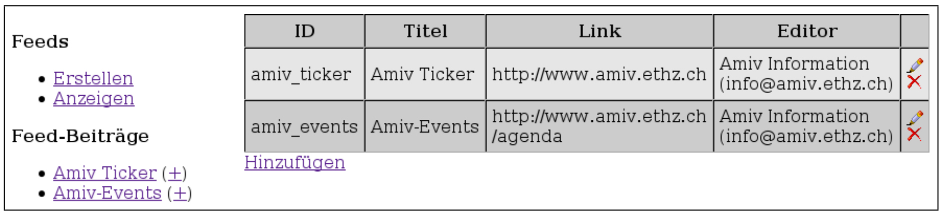
Feeds can be created, edited and deleted using the interface depicted in Fig. 5 and Fig. 6. The only data relevant are those listed in Tab. 5.

Table 8: Feed Content (infoscreen_feedContent)

Field name	**Data type**	**Description**
id	int(10)	Unique ID of the feed item
reference	int(10)	ID of the item in source table if the item has been imported. This is to prevent multiple imports of the same item
feed	varchar(30)	Name of the feed to which the item belongs
datetime	timestamp	Date/time indication for the item (e.g. an event)
location	varchar(100)	Location indication for the item
title	varchar(50)	Title
description	text	Description/content of the item
url	varchar(100)	URL indication for the item
image	varchar(150)	URL of an image to be displayed
active	int(1)	1 if the item is active and should be displayed, 0 otherwise
showDetails	int(1)	1 to show also the details, 0 to show only in calendar (if feed is mapped to calendar screen region). By this means, certain content items can be only listed on the calendar without the item's details showing up in another screen region, which has also subscribed to the same feed
created	timestamp	Creation time
updated	timestamp	Time of last update (this is set automatically)
showFrom	timestamp	Date/time from when to show the item
showUntil	timestamp	Date/time until when to show the item
priority	decimal(5,2)	A priority between 0.00 and 999.99

5.1.2 Content Management

Content can be managed easily through the web interface by selecting the name of a feed on the left side (clicking the + behind the name creates a new feed item for the respective feed). Feed items can be added, edited and deleted with



Feeds

- Erstellen
- Anzeigen

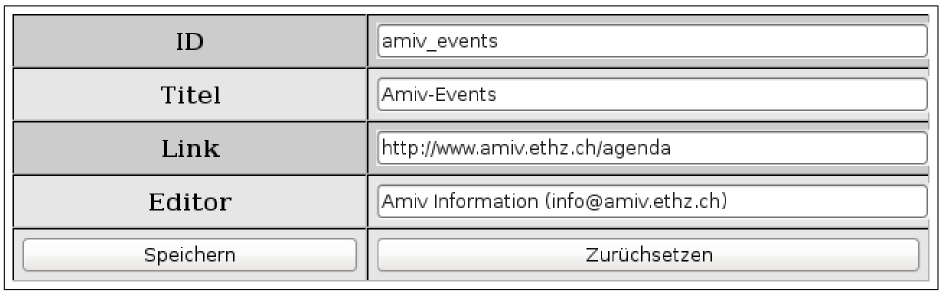
Feed-Beiträge

- Amiv Ticker (+)
- Amiv-Events (+)

ID	Titel	Link	Editor	
amiv_ticker	Amiv Ticker	http://www.amiv.ethz.ch	Amiv Information (info@amiv.ethz.ch)	✎ ✗
amiv_events	Amiv-Events	http://www.amiv.ethz.ch /agenda	Amiv Information (info@amiv.ethz.ch)	✎ ✗

Hinzufügen

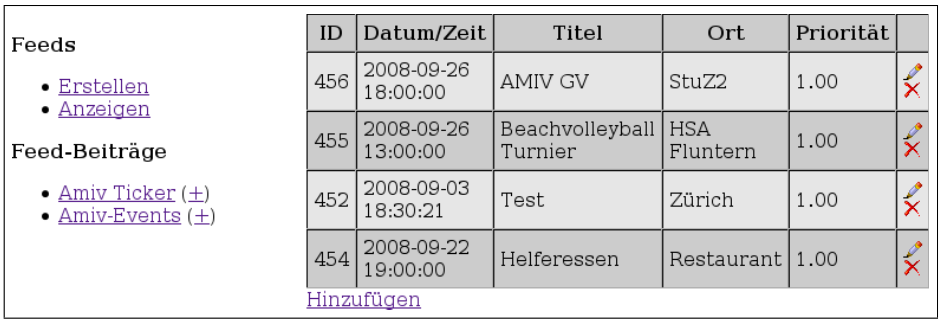
Figure 5: Feed Management Tool



ID	amiv_events
Titel	Amiv-Events
Link	http://www.amiv.ethz.ch/agenda
Editor	Amiv Information (info@amiv.ethz.ch)
Speichern	Zurüchsetzen

Figure 6: Feed Management Tool Details

a simple click. To display the detailed description for an item, simply click on its row and a description box will pop up beneath. For detailed information, please refer to the user manual in Appendix B. Fig. 7 and Fig. 8 show this tool.



Feeds

- Erstellen
- Anzeigen

Feed-Beiträge

- Amiv Ticker (+)
- Amiv-Events (+)

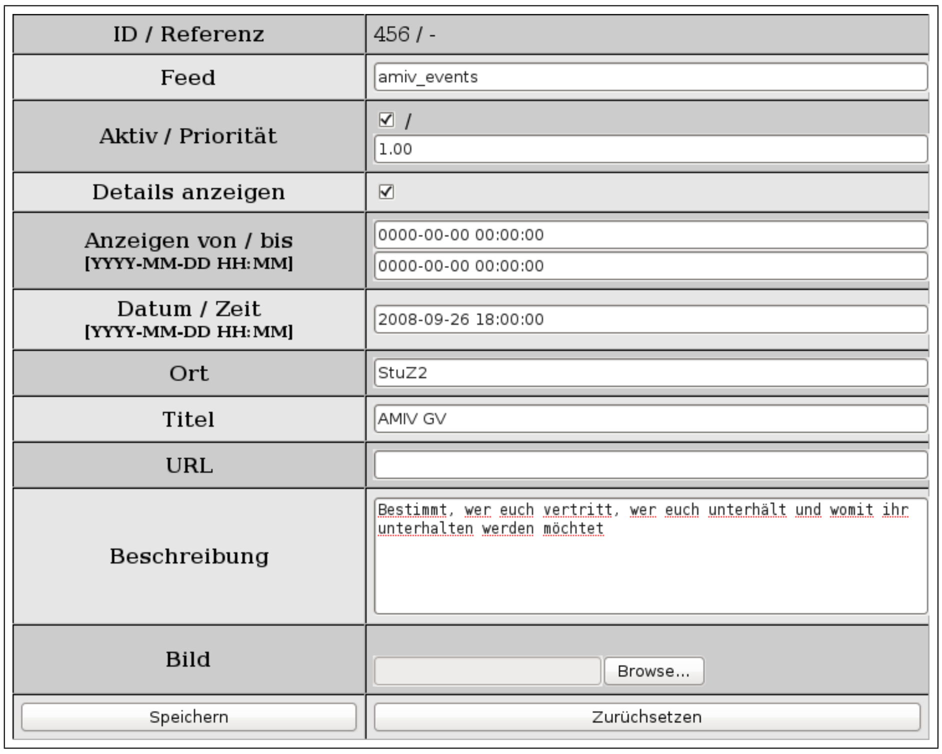
ID	Datum/Zeit	Titel	Ort	Priorität	
456	2008-09-26 18:00:00	AMIV GV	StuZ2	1.00	✎ ✗
455	2008-09-26 13:00:00	Beachvolleyball Turnier	HSA Fluntern	1.00	✎ ✗
452	2008-09-03 18:30:21	Test	Zürich	1.00	✎ ✗
454	2008-09-22 19:00:00	Helferessen	Restaurant	1.00	✎ ✗

Hinzufügen

Figure 7: Content Management Tool

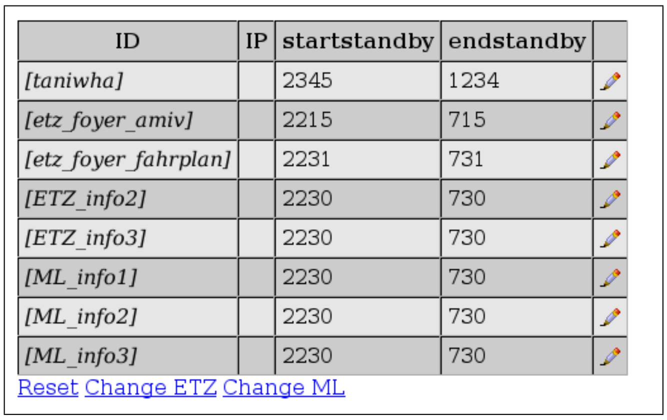
5.2 Standby Configuration

The configuration tool for the standby times is a very simple and easy to understand tool. It basically allows to set the times for the standby for each screen separately. One can also change the times for a whole set of screens, for example every screen in the ETZ Foyer. There is also a reset mechanism which allows to reset the times to their saved default value. Fig. 9 and Fig. 10 shows this config tool.



ID / Referenz	456 / -
Feed	amiv_events
Aktiv / Priorität	✓ / 1.00
Details anzeigen	✓
Anzeigen von / bis [YYYY-MM-DD HH:MM]	0000-00-00 00:00:00 / 0000-00-00 00:00:00
Datum / Zeit [YYYY-MM-DD HH:MM]	2008-09-26 18:00:00
Ort	StuZ2
Titel	AMIV GV
URL	
Beschreibung	Bestimmt, wer euch vertritt, wer euch unterhält und womit ihr unterhalten werden möchtet
Bild	Browse...
Speichern	Zurüchsetzen

Figure 8: Content Management Tool Details



ID	IP	startstandby	endstandby	
[taniwha]		2345	1234	✎
[etz_foyer_amiv]		2215	715	✎
[etz_foyer_fahrplan]		2231	731	✎
[ETZ_info2]		2230	730	✎
[ETZ_info3]		2230	730	✎
[ML_info1]		2230	730	✎
[ML_info2]		2230	730	✎
[ML_info3]		2230	730	✎

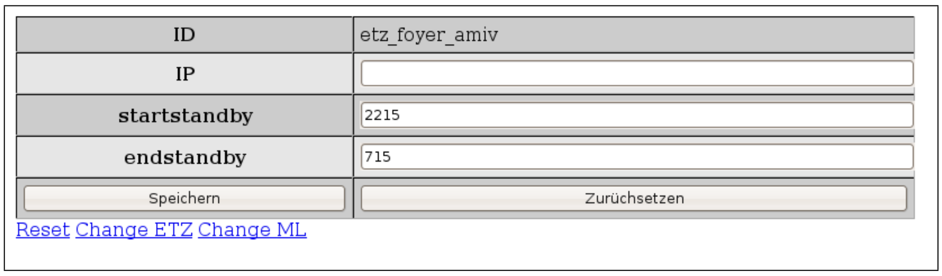
Reset Change ETZ Change ML

Figure 9: Standby Configuration Tool

6 PodNet Integration

The system was designed to be fully compatible with PodNet by taking the following measures:

- Information is introduced into the system in the form of RSS compatible



ID	etz_foyer_amiv
IP	
startstandby	2215
endstandby	715
Speichern	Zurüchsetzen

Reset Change ETZ Change ML

Figure 10: Standby Configuration Tool Details

feeds with the addition of some optional fields, such as time and location.

- The use of HTML as formatting solution was abandoned in order to ensure maximum compatibility and readability with mobile browsers.
- Images are to be loaded from a URI instead of directly including them into the corresponing RSS items.

Besides the points mentioned, the system was already compatible with PodNet by design. To distribute the PodNet feeds, a miniPCI WLAN card is attached on one board and the PodNet software is running on this system.

7 Comparison with existing systems

Information systems are quite popular and can therefore be seen at many places in the daily life. All these systems have different requirements, target different groups of people and use thus different approaches. To have a comparison to this system, two other systems are reviewed.

7.1 University of Zurich

The University of Zurich is running a system of screens on its whole campus. These screens can be found at many places such as foyers, mensae and libraries. The system is used to distribute all kinds of information such as upcoming seminars, readings or special lectures. Also the election of the students representation was advertised through this system and the results were announced. Despite this system, not many students took part at this election. And asked about these screens, they did not know about their existence.

7.1.1 Hardware

The system is built from a large plasma screen about 1 meter in diameter. Furthermore it contains a controlling unit to generate the picture. This unit is placed in a case at the backside of the screen. The size can only be estimated but it cannot be much larger than a standard ATX board.

7.1.2 Information System

The look of the information system is very smooth and well designed. It is built of a few nicely animated objects and is arranged as follows:

- At the top left corner, there is an animated logo of the university
- At the top right corner, there is a watch
- At the right side, there is an agenda. One item of this agenda is highlighted for about 20 seconds and the detailed information about this item is displayed in the main window
- At the left side, there is the main window with the detailed information

7.1.3 Pros and Cons

Pros:

- Large screen
- Nice animations
- Detailed informations

Cons:

- High power consumption
- Heat production that requires a fan (possibility for defects)
- First look is a bit messy

7.2 Main Railway Station Zurich

At the railway station there are mainly two different content types displayed. The most screens are used to provide current timetables. As such a system will be used as well at the ETZ building using an existing system from the VBZ[13] these timetables are not further regarded.
The second system is used to inform about problems on the railway system all over Switzerland and the resulting delays. As this system is completely dynamic, is can be compared to the AMIV system.

7.2.1 Hardware

As these screens are mounted at about 3.5 meter height. This leads to the need of a large plasma screen. Unfortunately, the controller hardware could not be inspected because of the height.

7.2.2 Information System

If there are no problems at the track system only a map of Switzerland can be seen and a text says that there is no problem. If there is a problem, a marker marks the location of the problem and a text box, connected to the marker by an arrow, informs about the problem and the effects on the passengers

[13]Verkehrsbetriebe Zürich

7.2.3 Pros and Cons

Pros:

- Large screen
- Well displayed information
- Simple but effective

Cons:

- High power consumption

7.3 Comparison

The comparison seems to be quite unfair because the AMIV system is compared against two professionally built systems. As the requirements and parameters differ, the comparison is possible though.
Advantages of the AMIV system:

- Low power consumption as a low power consuming board is used
- No fans used
- Standardized file format

Drawbacks of the AMIV system:

- Small screen
- No animations, because of low graphic power

8 Problems

8.1 Displays

The first display purchased was one from ASUS, Asus VW222U. Unfortunately, this display did not work properly with the alix1c board. It worked with every other computer without any problems. But connected at the alix1c, the resolution was `1280 x 1024` according to the OSD information. The strange thing was that even though the resolution was not suited for the display, the proportions of the images looked correct. For example the AMIV sign was perfectly shaped. At the login screen, there was a dotted line. On this line one could see some kind of beats. Several different configurations and drivers were tested with the Xserver without any progress. After testing the board on a different screen and making sure that the wanted resolution was possible, the most probable cause of the problem was a weak ADC[14] in the screen and a weak DAC[15] on the board respectively. This caused the screen to display the image incorrectly. To test several other screens, one board was taken to a store and 6 different screens were tested with permission of the staff. After this evaluation, a different screen was ordered which worked well.

[14]Analog to Digital Converter
[15]Digital to Analog Converter

8.2 Cursor

While testing the system it was discovered, that even with deactivated mouse (mouse device set to `/dev/null`), a cursor was displayed in the middle of the screen. There exist a few commands for the xserver config to fix this problem but GNOME ignored these. One further solution was to use a transparent cursor. Two different cursor packages with transparent cursors were found on the Internet. But it was not very simple to install them as one had absolutely no installation instructions and the other one was packed in a old installation packaging format. After several attempts it finally worked. The last thing to do was now to find the place the configuration was saved to be able to put this in the lockdown-script. Now a section in this script changes the entry in the corresponding file such that in the locked infoscreen, no cursor is seen.

9 Bibliography

References

[1] http://www.pcengines.ch/alix1c.htm

[2] http://www.samsung.com/us/consumer/detail/spec.do?group=computersperipherals &type=monitors&subtype=lcd&model_cd=LS22MEXSFV/XAA&fullspec=F

10 List of Tables

11 List of Figures

A Setting up the system

A.1 Interaction of System's Components

To be able to set up a system correctly, one needs to know how its components interact and how the role of each component is defined. To aid in understanding this, a short summary of each component together with a schematic of the system as a whole shall be provided here. Please note that the directivity of the arrows in Fig. 11 indicate the flow of information and are thus not random.

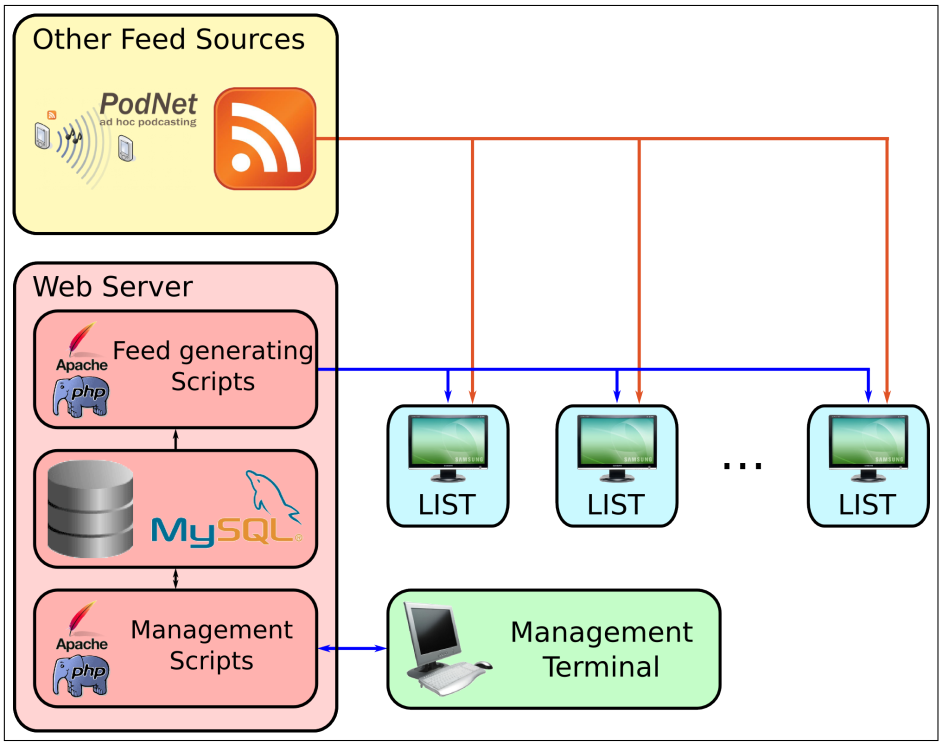


Figure 11: Interaction of System's Components

- The **MySQL Database** stores settings and content of self-managed feeds.
- The **Feed generating Scripts** turn the feed content and other information such as settings stored inside the database into RSS feeds.
- The **LIST Screens** access settings and feed content from the feed generating scripts and other feed sources, render and display them on an LCD display.
- The **Management Scripts** serve to change settings and add feed content to the database.

A.2 Display entities

The easiest way to set up the display entities is using the available ISO image with Debian Etch pre-installed. It is completely set up and configured. If for any reason, the image is not used, a few hints for the set up of the system are provided in the following sections.

A.3 Operating System

The operating system can be set up as usual. If using a flash card as storage, it is recomended to use ram disk for folders such as `/var/log` to prevent killing the card with unnecessary write cycles.

A.4 Gnome

Gnome has to be configured to automatically log in as the user used to run the applications. This setting is made inside the display manager (e.g. GDM). Furthermore, the standby script has to be automatically running when the user is logged in. First, the script `standby.php` needs to be stored somewhere on the system. Then, the user account needs to be configured to start the script. This can be done in the user configuration tool of gnome.

A.5 Security

For the security of the system, there are mainly two components. The first component is the firewall, which should be set up to disallow as much as possible. A sample configuration is provided in the file `firewallconfig.sh` which should be copied to the folder `/etc/init.d/`

The second component are the two scripts `openup` and `lockdown`. These two scripts should be copied to the folder `/usr/local/bin`.

A.6 Standby

To run the standby script, there are a few things to be done on the server.

- The two files `config.php` and `config.inc.php` need to be stored on the server.
- A table called `infoscreen_configs` needs to be created in the central database. (See Tab. 9)

A.7 Database / Web Tools

For storing feeds and settings, a MySQL database needs to be set up and it has to be reachable by means of network connection by all screens. The best way to do this is creating a new database account on a centrally managed server and running the SQL statements included in the file `list_setup.sql`.

To get the initial screen layout set up, please refer to section 4.1.1 for instructions.

The database location and credentials need to be entered in the file `database.inc.php` which resides on a web server together with all the settings and the management tools.

Table 9: Standby Configuration (infoscreen_configs)

Field name	Data type	Description
ID	varchar(30)	a descriptive ID containing the location and use of the screen (e.g. etz_foyer_amiv)
IP	varchar(15)	For future use. To automatically store the entity's IP adress
startstandby	int(4)	Time to start the standby (e.g. 2100 for 9 PM)
endstanby	int(4)	Time to end standby
defaultstart	int(4)	A default value for the start-standby. This value is used when the settings are reset
defaultend	int(4)	A default value for the end-standby. This value is used when the settings are reset

A.8 Firefox Extension

The extension is installed by simply opening the `.xpi` file inside Firefox and then entering the following settings in the extension's settings dialog (reachable via Tools → Add-ons → Infoscreen → Preferences).

- **Device ID**: A text string that represents a profile for the device. This can be unique, e.g. `etz_amiv_foyer` to describe just one device or non-unique to describe a class of devices. In the unique case, regular expressions can be used later on to select certain devices such as `etz_.*_foyer` for all devices in the ETZ Foyer.
- **Feed List URL**: URL of the script `feedList.php` that resides on the web server

B User Guide

B.1 Screen Regions & Feeds

To clarify these two terms, the concept of screen regions and feeds should be explained briefly:

Feeds are not much more than just a collection of "news items" in one file. Each item can have a title, some additional text and many more fields such as date and location. A feed is just a text file which is usually assembled from database content by a programmed script. On the internet, feeds are mostly advertised by giving the URL (address) of the corresponding script.

A **Screen Region** represents a certain area on the visible screen. Screens are usually subdevided into 2-5 independent regions, where each region can have its own content and type (e.g. calendar, ticker). What is displayed in a screen region is specified as a list of feed URLs (http://...) that indicate where to get the information from.

B.2 Management Tool

The management tool can be reached by entering the complete URL of the file `index.php` (on the web server) inside any web browser.

About all the tool's functionality lies in the buttons *add*, *edit* and *delete*. These operations can act on feeds, display exceptions, feed display settings and feed content; each of these areas is accessible by selecting the corresponding category in the left-hand navigation column.

The following should give an overview of what data are needed for the respective areas and which are the common links in between. This should be enough to administrate the system. If, however, more detailed/technical information is sought, section 5.1 and the tables in section 4.1 might prove helpful.

B.2.1 Feeds

Feeds need the following information:

- **ID**: A unique short ID that internally stands for the feed (e.g. `amiv_events`)
- **Title**: Name of the Feed
- **Link**: An optional Link URL where more information than just the feed content can be found, such as an organisation's web page
- **Editor**: Name and E-Mail address of the person responsible for the feed content

B.2.2 Display Exceptions

- **Rule Nr.**: A common integer number for all exceptions belonging together. So if there is a rule that treats every day of the week differently, there are 7 exceptions which carry the same rule number. This number is needed in the next section
- **Weekday**: On which day the exception should trigger
- **From, To**: Timestamps in the form *HH:MM* or *HH:MM:SS* indicating the time span during which the exception applies

B.2.3 Feed Display Settings

- **Feed URI**: Address of the feed or the script that generates it. If set up correctly, a drop-down menu can be used to select locally managed feeds
- **Active**: A feed can be disabled temporarily by removing this check mark
- **Priority**: A priority factor which is multiplied with each item's priority. This can be used to prioritize a feed as a whole
- **Exception Nr., show if exception doesn't apply**: The number of the exception that should be looked at (if any) and what is the standard behavior if the exception does not apply (show the feed or ignore it)

- **Display ID**: Select the screen region on which the feed should be displayed (see App. A for information on the set-up of screen regions)
- **Show on (RegExp)**: Regular Expression to specify on which screens the feed is to appear. This can be a screen ID in the form of a text string or a regular expression. To keep it simple, `.*` may be used to match any characters or `[^_]*` to match any characters except `_`.

B.2.4 Feed Content

- **Feed**: ID of the feed an item belongs to. This can be used to move items between feeds
- **Active**: An item can be disabled temporarily by removing this check mark
- **Priority**: A priority factor which is multiplied with the feed's priority. This can be used to prioritize individual items differently
- **Show from / until**: Used to limit the time span in which an item will be displayed. These fields take the form *YYYY-MM-DD HH:MM* and can also be left blank
- **Location**: An optional location to be displayed along the item (only on the MainScreen)
- **Title**: Title/heading of the item
- **URL**: An optional URL to be displayed along the item (only on the MainScreen)
- **Desription**: More text to describe the item in more detail (only on the MainScreen and TickerScreen)
- **Picture**: Allows uploading of one picture to be displayed on the left side of the item. If too large, this picture is scaled down to a maximum of 200 by 200 pixels while keeping the aspect ratio constant