Openlab - Automated testing and evaluation platform of the source code from assignments

Mihai Iachimovschi

Table of contents

List of Figures

List of Tables

1 Project Analysis and System Requirements

1.1 Project analysis

Openlab is a platform which aims to provide automated testing and evaluation of students based on their source code from the assignments. The goal of the system is to offer the possibility to organize the assignments for any subject in order to make them easily accessible for both students and teachers. It offers a transparent layer for setting strict deadlines, penalization for late submissions and even rewards for early submissions. Also, speaking of transparency, for any assignment there is a open grading policy which includes the requirements list and the points a student can achieve. Thus, it is intended to improve the efficiency coefficient of students and professors.

The system has a web interface which makes it extremely flexible and cross-platform. The UI is simplified as much as possible so it can be used by anyone without additional training. From the student's perspective, the system will run on his code a prepared test case that was written by the professor and will publish it's results. All the external code should be executed in a completely isolated environment in order to protect the server from malicious code that can be submitted.

Conceptually speaking, the core of Openlab can be described as a platform which can accommodate fully isolated containers that compile and execute all the provided source code, saving the results to a database.

1.1.1 Problem description

Students from the IT faculty are dealing with a lot of laboratory works, homeworks and individual assignments for which they should implement different algorithms by writing small programs. For each subject, the requirements are presented in varying forms which makes a inconsistent work-flow for both students and teachers.

Scheduled laboratory sessions become focused on verification and evaluation of students' past assignments. This activity is very time consuming for the professor and it turns out that almost all the time reserved for the lab is consumed by this important but time-wasting procedure.

Automated testing and evaluation of the source code can obviously improve the learning process by focusing on more important and useful tasks. Professors can spend more time on explaining different approaches, technologies or algorithms that can be used for the assignments instead of verifying past assignments in the time reserved for the lab class.

People tend to procrastinate, that's why a lot of students submit their assignments late. Not having a obvious overview of the deadlines can be misleading for students. Being confused may leave

space for excuses. That's why enforcing transparent deadlines,notifications and bonuses for early submissions may improve student's punctuality which is useful not only in university.

1.1.2 Overview of similar products

Most of the similar projects position themselves as LMS (Learning Management System). The main goal of learning management systems is the delivery of electronic educational materials to the students. It offers also to the professors a way to administer and document courses and also track students' results by evaluating assignments and tests.

Moodle is a flexible, free software, open-source learning platform. It is a LMS that offers basic assignment submission features, forum for discussions, glossary of definitions, a taxonomy for courses and lectures and on-line quiz module. Moodle is written in PHP and can be deployed on any server capable of running PHP and it requires a web server such as Apache or NGiNX. It is compatible with MySQL, PostgreSQL and other relational database management systems.

DigitalChalk is a proprietary Learning Management System which allow users to create and deliver their courses materials on-line. The solution is provided to the customers as a SaaS (Software as a Service). The price starts at 4.95$ per month per user with an initial setup fee starting at 399$.

Blackboard Learning System is a complex proprietary Learning Management System. It is highly customizable and it can be either deployed on local server or used as SaaS. The company's pricing policy is not publicly available, however it is rated as expensive.

All considered applications are great tools, highly functional and useful, but none of them has the possibility of automatic testing and evaluation of source code for the assignments which is very important for IT faculty.

Some of the similar products offer too much overhead in terms of unneeded functional while others are too expensive for a non-profit organization. Openlab aims to provide a minimalistic and simplistic approach to the problem solving. It is also Free and Open Source Software.

1.2 System requirements

1.2.1 Product functions

This section provides a brief overview of the functions that system will perform. Below are presented the key functions of the product which must be implemented in the final version. The whole system should have the following features:

– Keep track of laboratory assignments for users (students);

– System should accept assignment submissions and evaluate them;
– System should provide an interface for professors for adding new assignments and test cases;
– Execute safely the submitted code and the test and save the results;
– Grade automatically the submitted and executed assignments;
– Inform students about last changes via a private feed;
– Display upcoming assignments for logged in student;
– Display upcoming laboratory sessions;
– Display enrolled courses that registered in the system;
– Keep track of grades from the automatically graded assignments;

1.2.2 Constraints and dependencies

This sections presents basic requirements for the whole ecosystem. A client that uses the applications has to have a modern web browser installed on his system. On the back-end side, constraints and dependencies are the following:

– Server should run any Linux distribution with kernel newer than 3.10;
– Docker 1.4 or newer should be installed;
– Python 2.7 or newer should be installed;
– Python PIP should be installed. It will provide packages as: fig, virtualenv, django, etc.;
– TCP ports :80 and :443 should be available;

1.2.3 User questions for information system

The user interface represents the space by which user interacts with the system in order to benefit from system's features. It should provide instant feedback for user's actions. It should also give us the answers of predefined questions:

– What are the upcoming assignments?
– What are the deadlines for the future assignments?
– For when is scheduled the next laboratory session?
– What are the enrolled courses?
– What is the amount of credits obtained by a specific subject?
– What are the grades for specific subject?
– What are the requirements of specific assignment?
– Which are the results of my previous submissions?
– How many submission trials are left?

– What is the highest score obtained for a specific assignment?

1.2.4 User interface requirements

This section presents the description of the interface, which should create the bigger picture of how the platform should look like. The main goal is to keep interface as clean and minimalistic as possible. The three main views can be seen in: Figure 1, Figure 2 and Figure 3.

For best results we should take into account these aspects:

– All visual elements that provide key functionalities should be easily accessible;
– The User Interface should have a steep learning curve;
– The interface should avoid ambiguity by making everything clear with text or graphical elements;
– Responsiveness should be guaranteed;
– The interface should be engineered in a forgiving way such that the user should be better notified about the potential mistake rather than punished;
– The interface should not be bloated with irrelevant nor redundant elements.

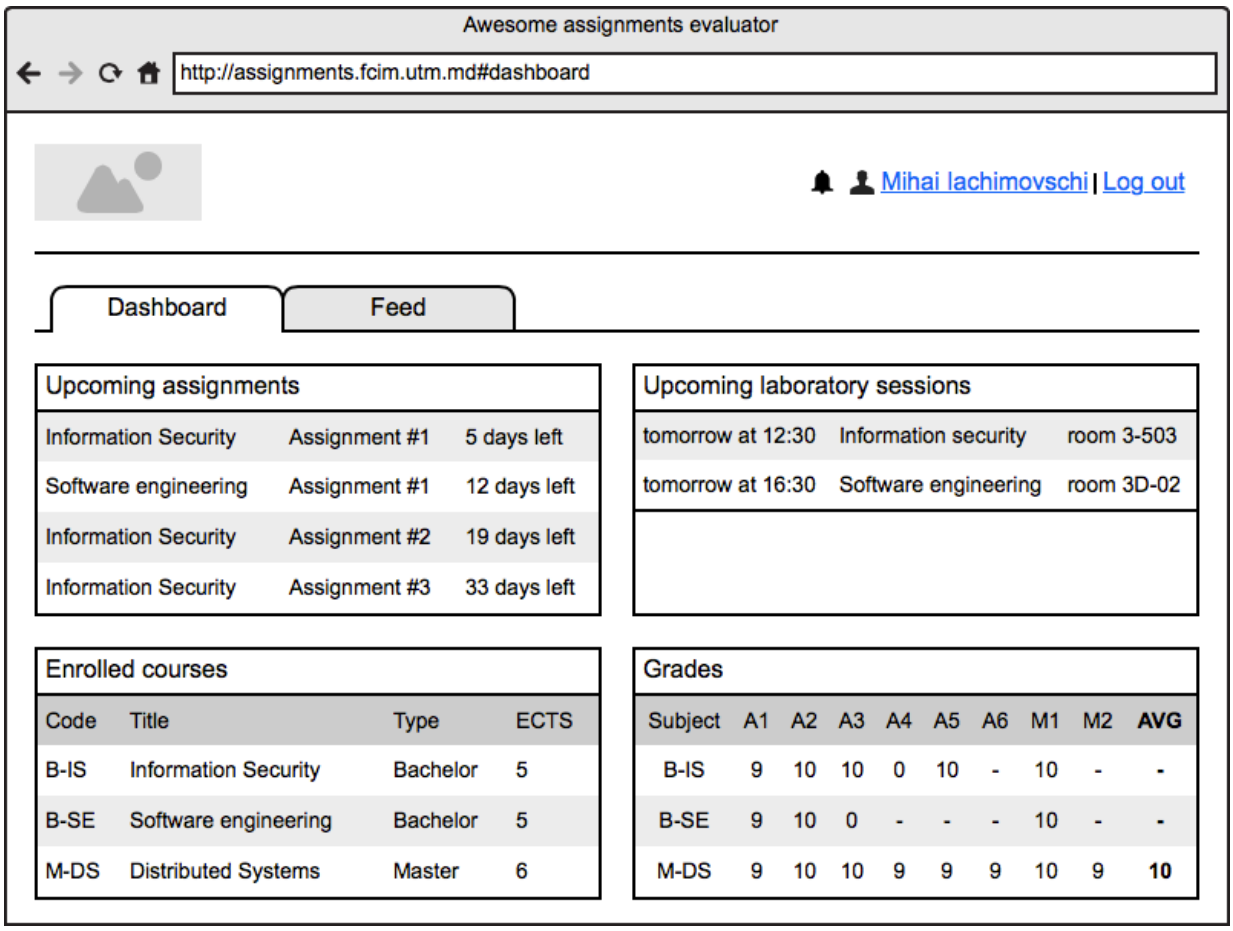

Figure 1: Graphical User Interface mock-up for the dashboard page.

In Figure 1 is represented the page with the user's dashboard. The content from this page is split in four regions that makes an overview of the current state of the user. The four zones provide to the user information about: upcoming assignments, upcoming laboratory sessions, enrolled courses and obtained grades.

This page has the purpose of giving an informative snapshot of the state of student in his relation with the Openlab system. It offers an easy way of visualization of all the details organized on a single page which aims to help the user to make decisions regarding his time-management and also to evaluate the past productivity level.

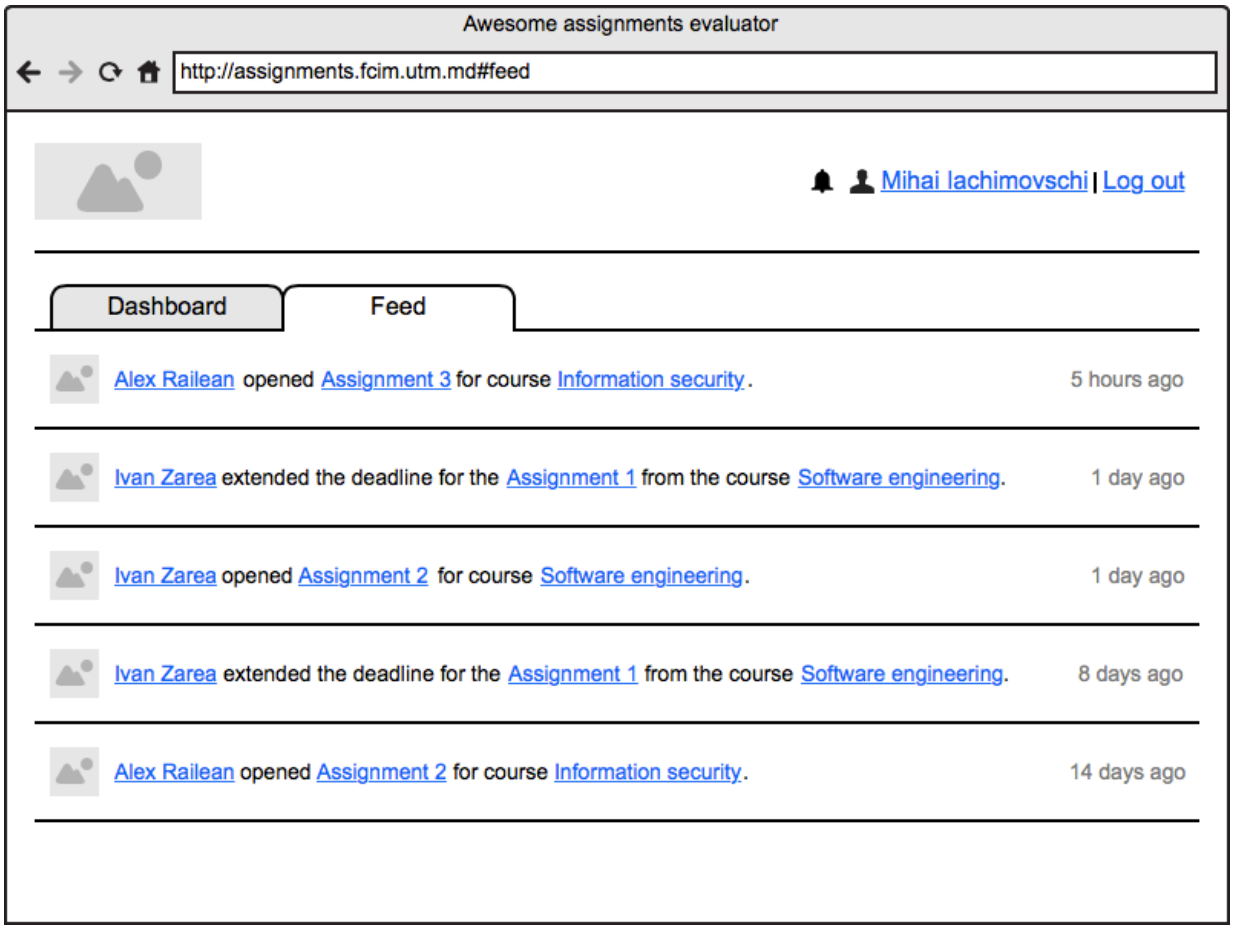

Figure 2: Graphical User Interface mock-up for the feed page.

Figure 2 represents the user interface of the feed page. The content of this page is organized in a simple and minimalistic way, taking advantage of white-space. On this view user can find a relevant overview of the activity regarding his enrolled courses. The list of past events is sorted by the date of event, so the most fresh one is on the top of the list. Also, the user have access to navigate directly from the feed to the specified course, assignment or professor's page by hyper-links.

Figure 3 displays the most functional-dependent page that provide useful information and also has a call to action which provides to the user the possibility to submit his work. The view is designed such that it provides the detailed information about the assignment requirements and this

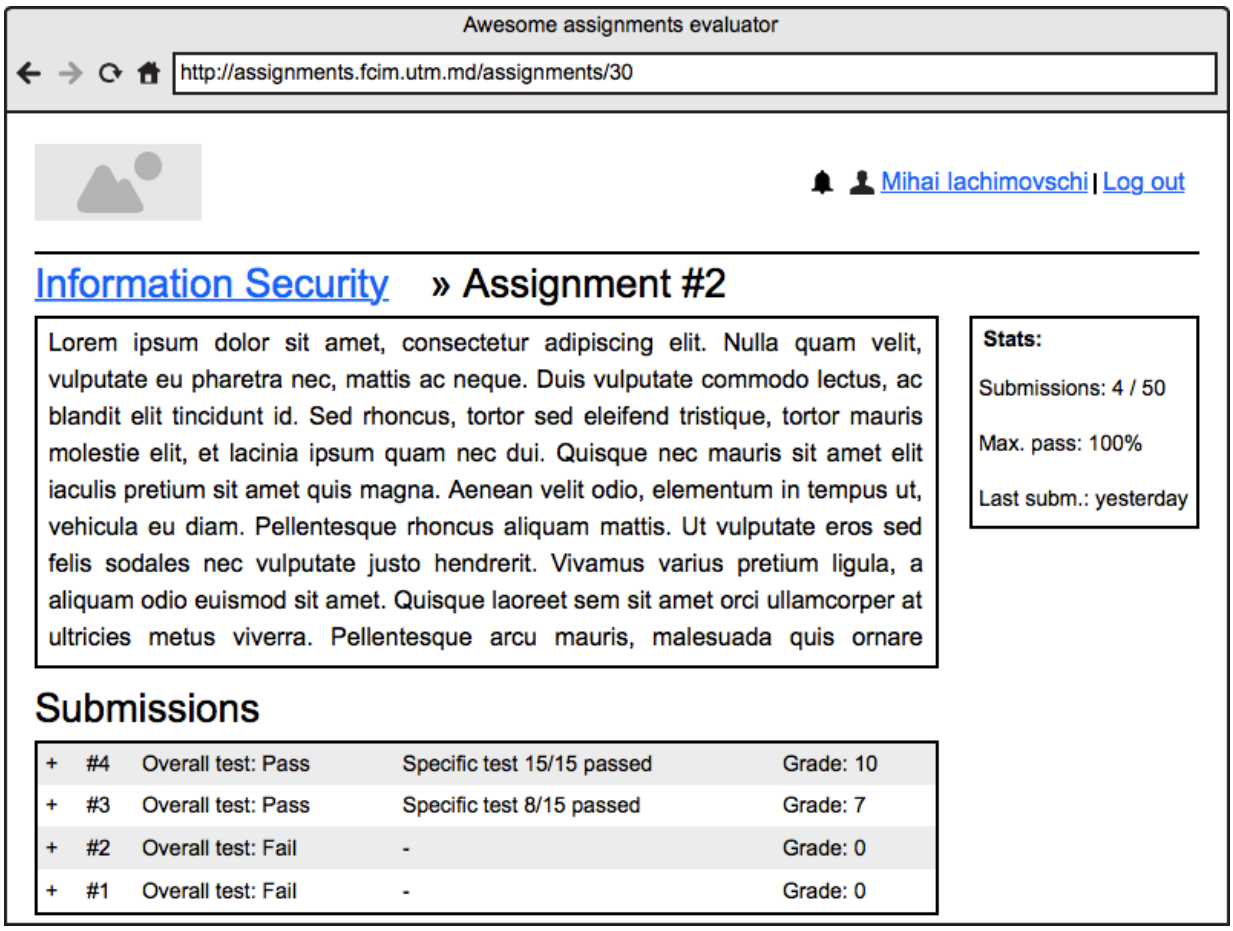

Figure 3: Graphical User Interface mock-up for the assignment page.

piece of information occupies the most of the space. Also, there is a widget with brief statistics about user's success and progress on current assignment. There is also a section which gives us detailed information about each past submission – visible on user's action.

2 Software Modeling

2.1 Deployment

Software deployment is the set of actions that are performed in order to make the system available to the end-user. Deployment consists of activities that may be interrelated or repetitive which are be performed sequentially. Considering the fact that each and every software system is uniquely engineered, the deployment process can be only defined as a general procedure which is distinct from system to system.

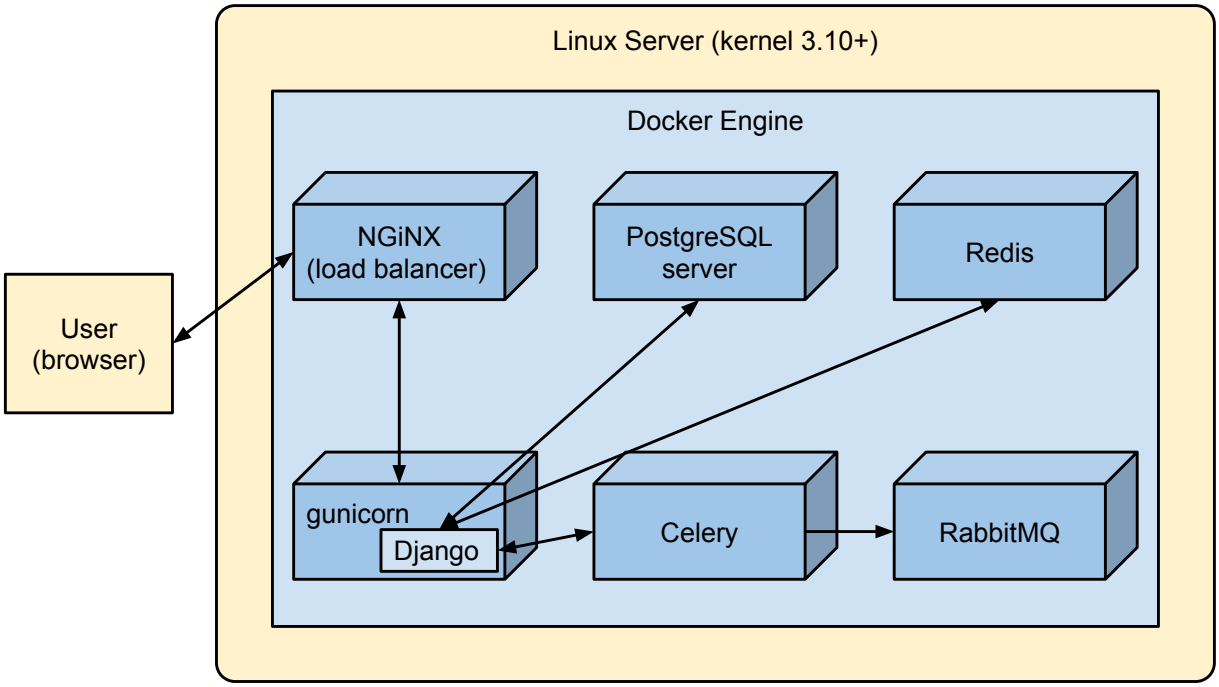

Figure 4: Deployment of isolated components using Docker containers.

In figure 4 is represented the general overview of the isolated services that are able to communicate between each other. The user is able to communicate directly only with one of the containers, the web server which acts like a reverse proxy between the client and the wsgi server.

2.1.1 Install

All the components could be orchestrated using Fig utility which aims to create work-flows as described above by a unique declarative configuration. By the time of writing this paper, Fig tool was adopted by Docker and re-branded as Docker compose.

```
1 web:
2   build: .
3   command: python app.py
4   ports:
```

```
    - "8000:8000"
  volumes:
    - .:/code
  links:
    - redis
    - postgress

redis:
  image: redis

postgress:
  image: postgress
```

This snippet of code is the declarative configuration for Docker compose which creates three containers that are executed. The logs generated by them are combined into one single flow of messages, which are marked properly, so are distinguishable. The `web` container has two links to `redis` and `postgress` respectively. The links permit them to share data with each other and at the same time these containers are fully isolated from the outside world. The `redis` and `postgress` instances have default configuration which could be altered to match all the needs.

The written code is not the only piece of software that runs in the `web` container. It is written under the Django framework so it should have all it's components already installed into the container. Fortunately the procedure of getting things done is extremely simple, just installing in the container the PyPy packages from `requirements.txt` by issuing `pip install` on them. All other configuration files are already present in the directory with source code. Some of the configuration can be altered by changing the Docker compose yaml configuration file and specifying required environment variables that should be overridden.

2.1.2 Update

Even though the install process seems quite easy and straightforward, the update process is a little bit trickier. The simple update of the container's contents is similar with the initialization described in previous section, but also needs some additional actions that alters the database, clears the caches and so on.

A better approach is a zero-downtime update that could be implemented with a little bit more effort. The idea is to create a new working instance and telling the web server to point to the new instance such that the user will not be aware of the process that just happened.

Figure 5 describes the work-flow of the zero-downtime update. As the NGiNX is capable of handling multiple upstream servers, it can clearly act as a load balancer. That means that it can point to multiple upstream servers and balance the load between them in a uniformly distributed

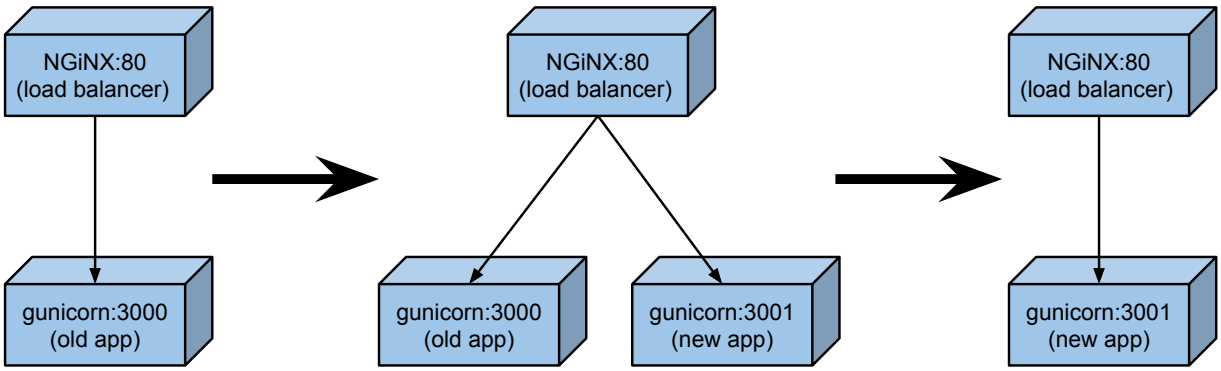

Figure 5: Update the application with a zero-downtime.

manner. If one of the servers is down, it will automatically try to forward the request to the next one and it will fail after all servers are down. In that way there is a possibility of seamless update by creating a new working instance, updating the load-balancer, softly killing the previous instance, which means that it will respond to all requests but it will not accept new ones. The whole process can be described by the following steps:

1. Update the docker image;
2. Creating a new, working, updated instance;
3. Update the upstream configuration;
4. Softly killing the old instance;
5. Updating the upstream configuration once again.

3 System Implementation

3.1 Web platform development

3.1.1 Internationalization and localization

The application's interface should be translated to few languages so there is an imperative need of internationalization by defining translation strings and eventually translating them. Django internals offer to the developer all the required tools for this procedure. The built-in functionality relies on the GNU gettext tool-chain.

Internationalization (i18n) is the process of preparing the software solution for localization by specifying the translatable strings, plurals, and all locale dependent fragments which will be then exported to message files that are then given to translators for the process of **Localization** (l10n). Django offers the necessary tool for extraction of translation strings to message file `.po`. These files can be edited by the translators with special editors like Poedit. To accomplish this, the translated strings should be marked as follows:

```
1 from django.utils.translation import ugettext as _
2 _("String that needs translation")
```

Message files in Django are automatically generated issuing the command `django-admin.py makemessages`. After successful generation of the messages files it is possible to compile them into binary `.mo` files such that in the end the directory structure looks like this:

```
po
├── cs_CZ
│   └── LC_MESSAGES
│       ├── file.mo
│       └── file.po
└── en_US
    └── LC_MESSAGES
        ├── file.mo
        └── file.po
```

The generated `.po` files have a extremely minimalistic layout so it can be altered by a person which is not at all initiated into programming. These files can be also edited with special software like Poedit. The example syntax of such a file is presented here:

```
1 # path/to/python/file.py:3
2 msgid "String that needs translation"
3 msgstr "Textový řetězec který potřebuje překlad"
```

From the developer's point of view, the i18n mechanism provides a reliable way to make software capable to be translated to different languages thus making it accessible for people around the world. The job of localization (l10n) can be easily delegated to the competent translators that don't mind coding.