Department of Computer Science,
Electrical and Space Engineering

BlueHive (www.bluehive.me)
https://github.com/ThibaultR/BlueHive
Design of Dynamic Web Systems

Christoph **Malin**
Thibault **Rapin**
winter semester 2015

14.01.2016

Design of Dynamic Web Systems (M7011E)

Table of contents

Idea

BlueHive is an online platform used by companies for managing events. For example catering companies often have irregular events and employees with irregular schedules, sometimes students or people with part time jobs. The companies want a fast and efficient way to manage events and the users working at the events.

The company/administrator should have the ability to

- Manage events (add, change)
- Manage users working for an event
- Manage users
    - activate new users
    - deactivate users
    - change user accounts (user data, password, profile picture)
- Manage user groups (add, change)

The employees should have the ability to

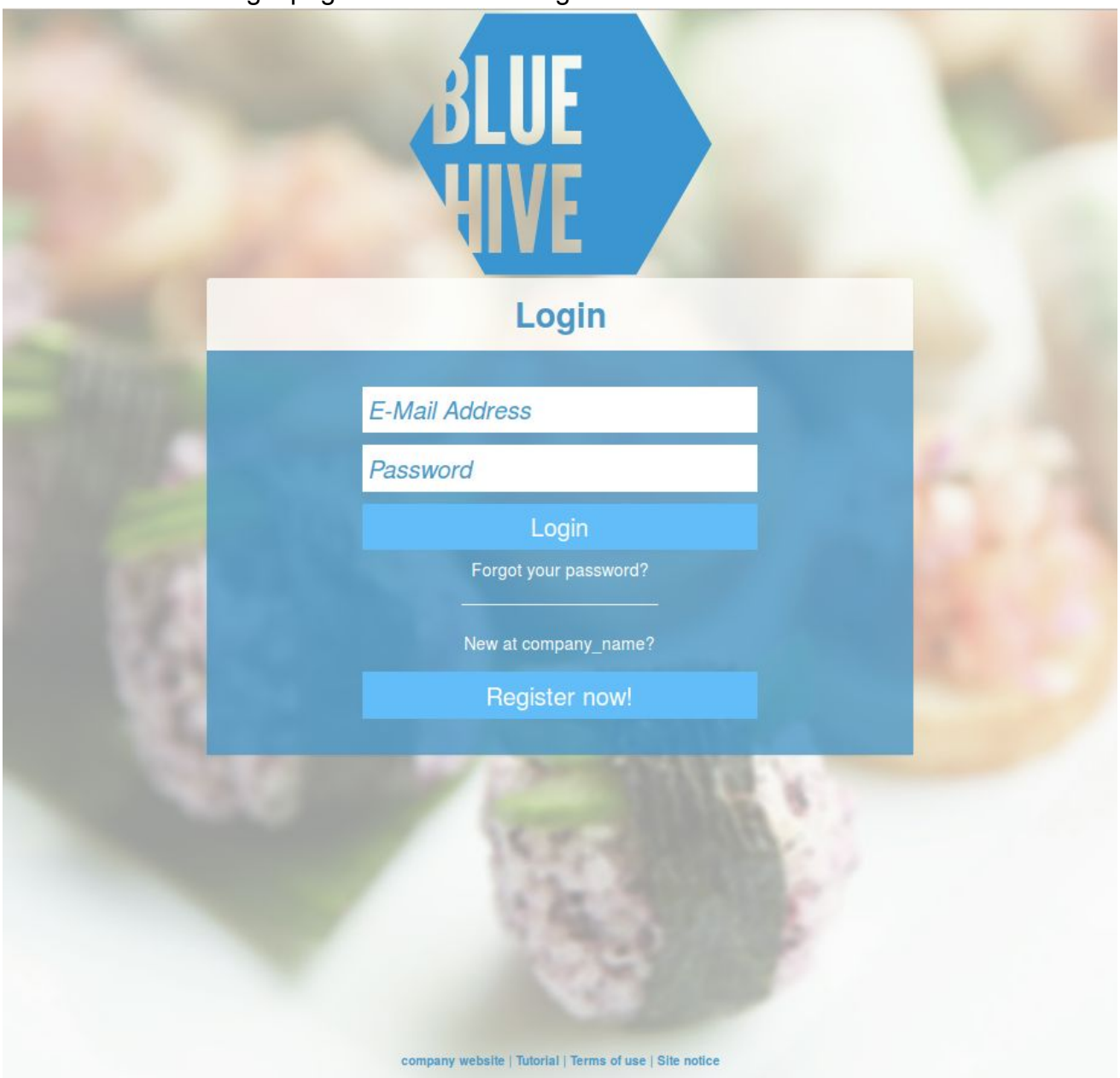
- Register with personal data and picture upload
- Change personal data, password and profile picture
- See all available new events
- Apply for an event
- See the status of applied events

User Scenario

The following scenario shows screenshots of the implementation and how the users can see and use the BlueHive system.
The website is reachable under the URL http://www.bluehive.me.

Start Page

The user sees the login page. There he can log in with his credentials.



Registration

If the user is new he needs to register first. Therefore he clicks on the **Register now!** button. At the registration page he needs to upload a profile picture and enter his personal data.

When the registration is successful, the administrator sees that a new user registered. The user is not able to login at this point because the administrator needs to check the application first and can choose to activate or delete the user.

User Events

When the user is logged in he sees all events he already applied to (My events). Below he sees all new events available for him. The colors in the My events part tell the user the status of his requests. Green means that the user is accepted, red that he is rejected and blue that the status is open (the administrator didn't decide yet).

User Profile

The user has also the possibility to change his profile data. Therefore he clicks on the Profile link when he is logged in. Now he can change his profile picture, his password and all other data he provided during the registration.

Administrator Scenario

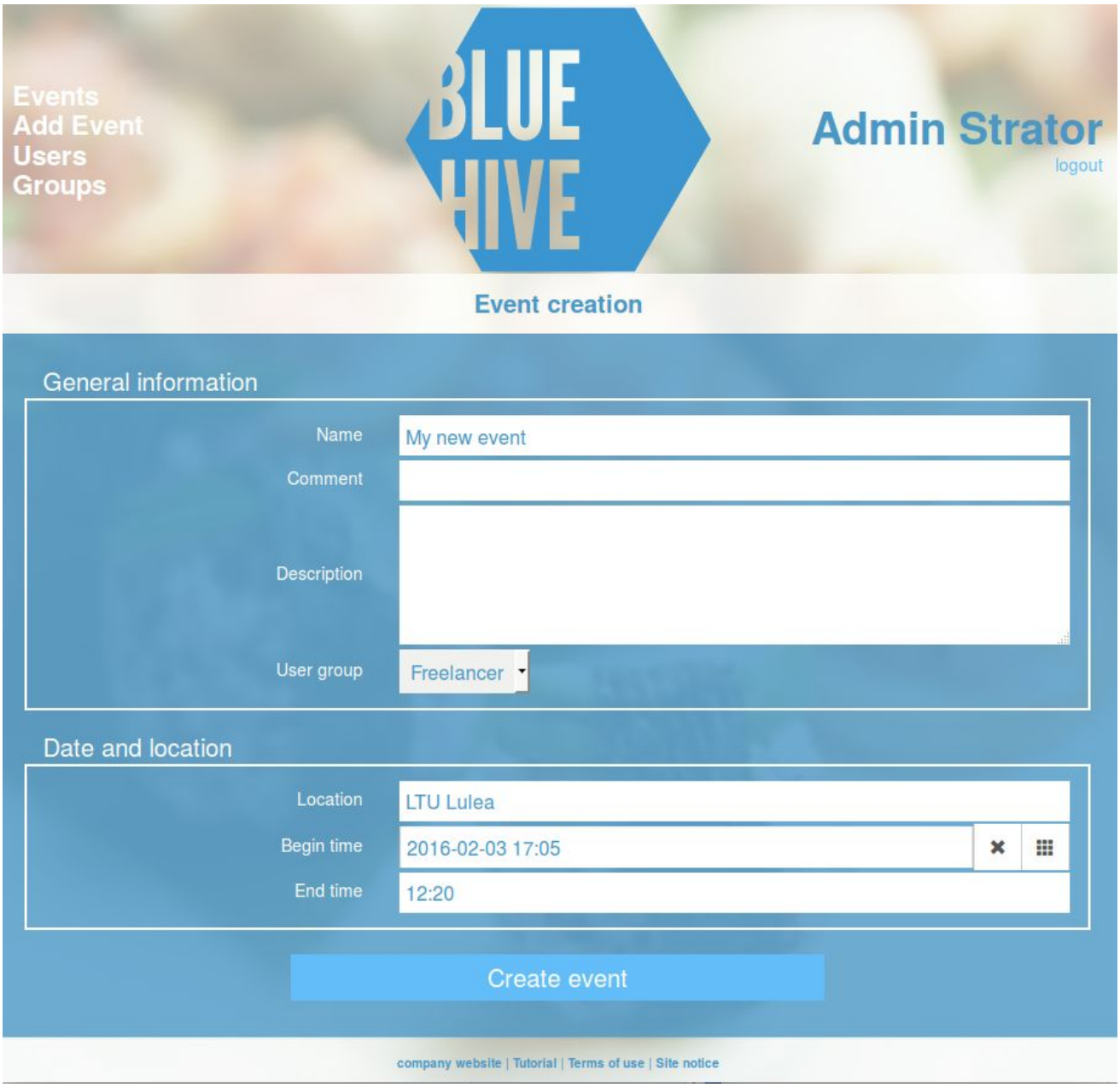
The following scenario shows screenshots of the implementation and how the administrator can see and use the BlueHive system. He uses the same login page as the users. The system recognizes that an account has administrator privileges and provides a different view to the administrator.

Administrator Events

The administrator sees all events starting yesterday or later. When he clicks on an event he sees the users who applied for the event, the comments they wrote and their ratings. He can accept/reject users by simply clicking on the accept/reject button.

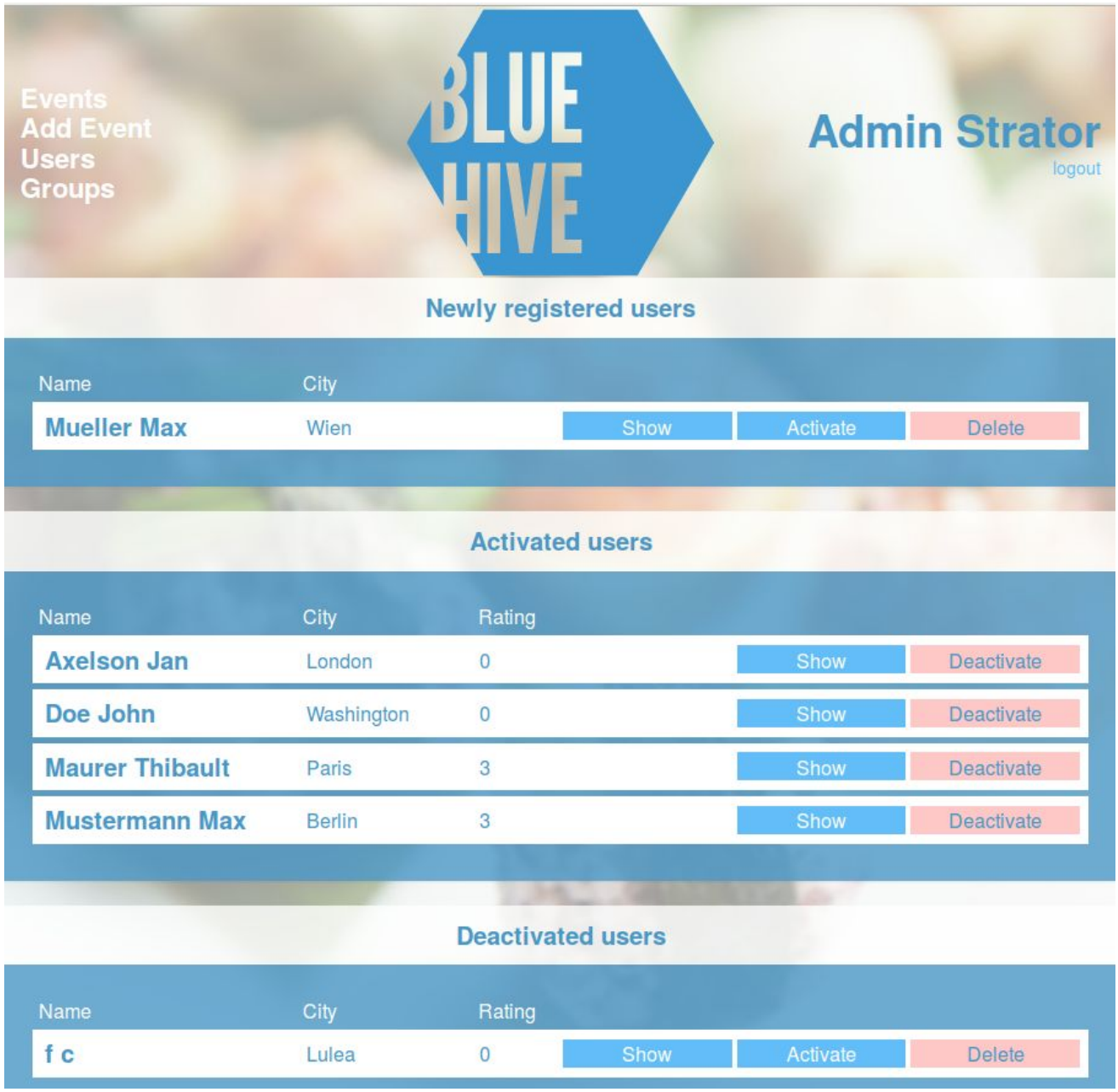
Add an event

The administrator can create an event with a specific name. It is important that he sets the user group. Every user is member of one or more user groups. An event is only visible by users belonging to the selected group.
The administrator can also set the location, begin time and end time.



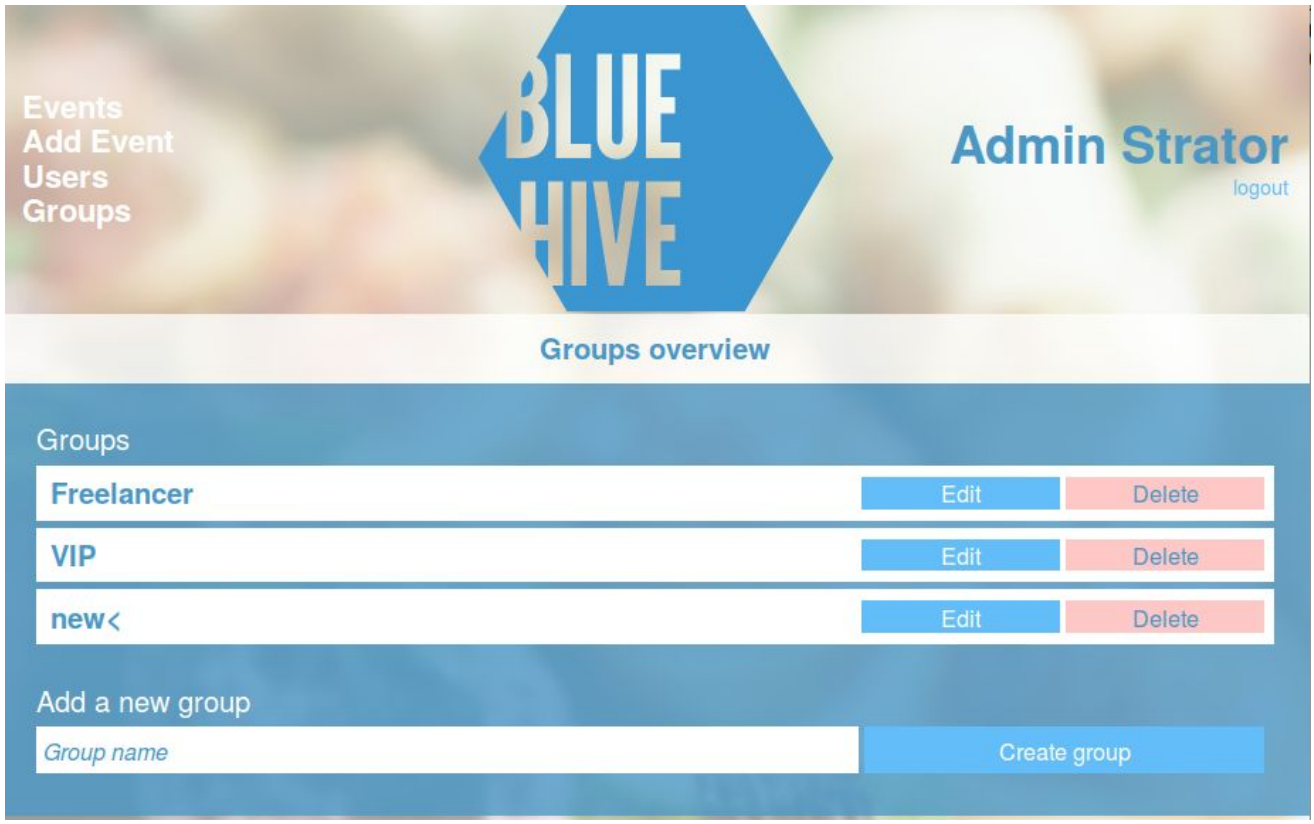
Manage users

The administrator can change the status of the users (activate, deactivate, delete). He can also look at their profiles and update their data.



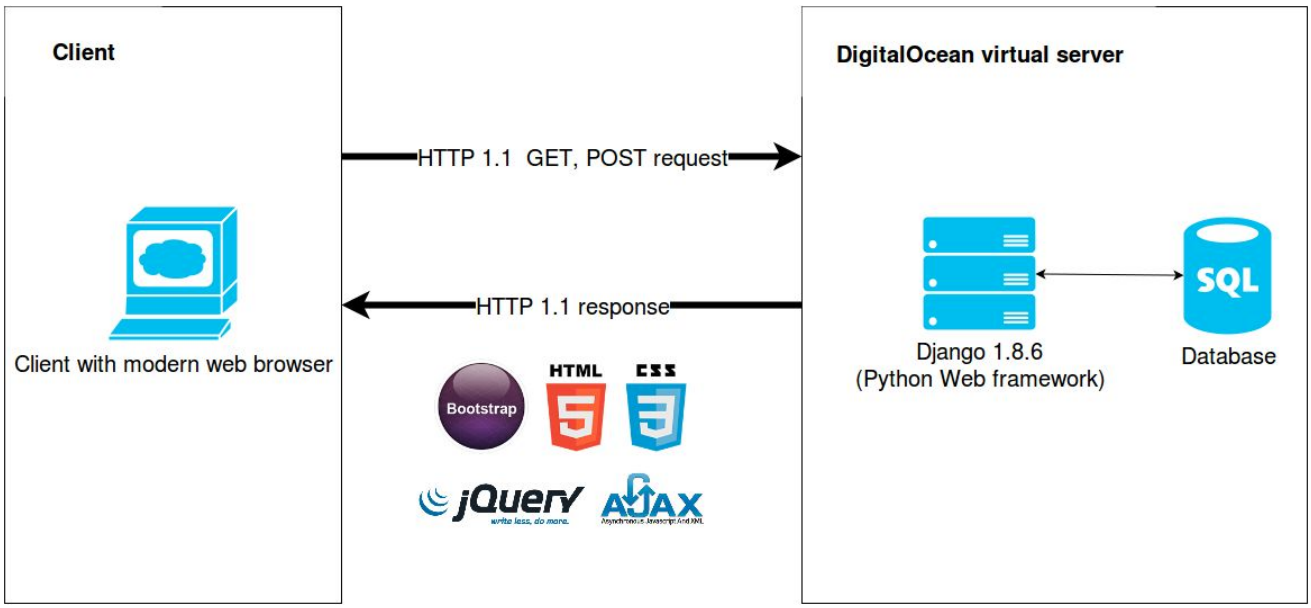
Groups

The groups view allows the administrator to create new groups or delete/rename already existing groups.



Architecture & Technologies

The following diagram shows the architecture of BlueHive and the technologies we used to implement and also to deploy it.



We wanted to use a technology which was new for us and also is nice to know for the future. Since we already know and don't really like JavaScript and PHP we chose python as preferred language. A big advantage of python is that it is fast, easy to write and very popular. After this we had to find a decent framework which fits our requirements. We chose the django framework over flask.

Django has the big advantage that it already provides a lot basic features like session management, templates, message delivery. Very important was also that there exists a good documentation and a very active community helping us when problems occur.

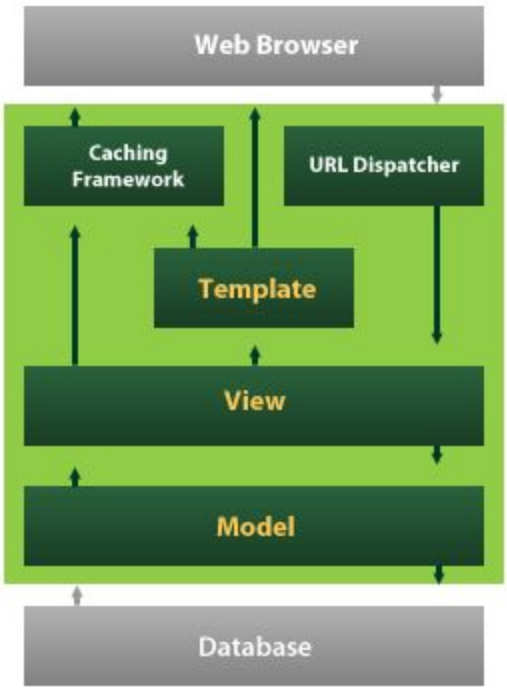
Another big advantage of Django is that the database can be easily changed between SQLite, PostgreSQL, Oracle and MySQL. At the moment we use SQLite because it is easy to create, modify and backup.

For the frontend we use HTML5 and CSS3. Twitter's bootstrap framework helps us with developing a responsive website. Bootstrap is fast, easy to use and has a really active community. To implement a dynamic and flexible website we use jQuery and AJAX.

The diagram shows the client on the left side sending an HTTP request to our server. The server processes the request and sends a response back to the client.

Interaction between software parts

The following graphic gives a basic overview about how django works.



1. The URL dispatcher (urls.py) maps the requested URL to a view function and calls it. If caching is enabled, the view function can check to see if a cached version of the page exists and bypass all further steps, returning the cached version, instead.

2. The view function (usually in views.py) performs the requested action, which typically involves reading or writing to the database. It may include other tasks, as well.

3. The model (usually in models.py) defines the data in Python and interacts with it. Those data can be easily stored in a MySQL, PostgreSQL or SQLite database.

4. After performing any requested tasks, the view returns an HTTP response object (usually after passing the data through a template) to the web browser.

5. Templates typically return HTML pages. The Django template language offers HTML authors a simple-to-learn syntax while providing all the power needed for presentation logic.

API

We provide an API for other developers to generate an android app for example. Due to security reasons we allow only read access to our API at the moment.

For creating the API we had to choose between the Django REST Framework and Tastypie (http://tastypieapi.org/) .
We decided to use Tastypie because it is easier to understand and it looks like more people are using it.

Authentication

To access data from the API you need a valid username API-Token combination. The API-Token of a user gets generated together with a new user account. At the moment every user + API Token has read-only access to all other user events. Of Course this is not how privacy should look like in the future. For the moment we decided that the Token is not visible to the users. The server administrator must extract it out ot the database and send it to the respective people.

Requests are only answered when they contain following security data:

```
# As a header
# Format is ``Authorization: ApiKey <username>:<api_key>
Authorization: ApiKey daniel:204db7bcfafb2deb7506b89eb3b9b715b09905c8

# As GET params
http://127.0.0.1:8000/api/v1/entries/?username=daniel&api_key=204db7bcfafb2deb7506b89eb3b9b715b09905c8
```

We decided to use JSON as format for data communication.
The basic urls available are

- http://www.bluehive.me/api/v1/user
- http://www.bluehive.me/api/v1/event
- http://www.bluehive.me/api/v1/event_request

If you add “/schema” at the end of each URL you get the Schema of that resource.
To get all data of a specific event you need to send the following GET HTTP Request:

```
http://www.bluehive.me/api/v1/event/1/?username=admin@example.com&api_key=e63459fe07d7c8076c45cce650b33c081a2264a3

Response:
{"begin_time": "2016-11-22T22:30:59", "comment": "only for important ", "date_altered": "2015-11-25T20:25:16.257331", "date_created": "2015-11-25T20:25:16.257306", "description": "", "end_time": "two days later", "id": 1, "location": "Vienna", "name": "G8 Meeting", "resource_uri": "", "status": 0}
```

More infos about using Tastypie:
http://django-tastypie.readthedocs.org/en/latest/tutorial.html

To test the API we found the python programm pyresttest (https://github.com/svanoort/pyresttest) . It allows to test our API very easily and shows if everything works as expected. After installing the program you can create a text file called **bluehivetest.yaml** for example. Put the following code into the file:

```
---
- config:
        - testset: "BlueHive app tests"

- test:
```

```
        - group: "BlueHive"
        - name: "Get all users"
        - url:
"/api/v1/user/?username=admin@example.com&api_key=e63459fe07d7c8076c45cce650
b33c081a2264a3"
        - method: "GET"

- test:
        - group: "BlueHive"
        - name: "Get al specific user"
        - url:
"/api/v1/user/1/?username=admin@example.com&api_key=e63459fe07d7c8076c45cce6
50b33c081a2264a3"
        - method: "GET"

- test:
        - group: "BlueHive"
        - name: "Get event request of a specific user"
        - url:
"/api/v1/event_request/?user_id=2&username=admin@example.com&api_key=e63459fe
07d7c8076c45cce650b33c081a2264a3"
        - method: "GET"

- test:
        - group: "BlueHive"
        - name: "Get a specific event"
        - url:
"/api/v1/event/1/?username=admin@example.com&api_key=e63459fe07d7c8076c45cce
650b33c081a2264a3"
        - method: "GET"
```

To run this config file you need the following command:

 # resttest.py http://www.bluehive.me bluehivetest.yaml

Output:
Test Group BlueHive SUCCEEDED: 4/4 Tests Passed!

Security

One big advantage of the usage of the django framework is that the developers already thought about security in different parts of the application. Of course the default security is not enough but it helps beginners to avoid very dangerous security risks.

The following list shows different parts of the web application which are already secured by django.

- **SQL injection**
  - Unlikely cause of the usage of the django database-abstraction API. The developers already assured that it is not possible to inject code into an sql query.
- **Password Storage**
  - Password algorithm recommended by NIST with a SHA256 hash.
- **Session Management (part of the framework)**
  - Changing password logs out all opened sessions of the user.
- **Cross site request forgery (CSRF)**
  - CSRF token needed for every POST-REQUEST.

There are much more things to consider in the future e.g. OWASP Top 10.

Discussion and evaluation

As shown in the administrator and user scenario all our desired features were implemented in time. Some functionalities are not perfect yet but the project is on a good way. This was only possible due to the use of github.com for code revisioning and IntelliJ Idea as an development environment.

Future work

We are really enthusiastic about our project and hope to continue the work. The basic system works and can be made to something beautiful with multiple small changes. The following list gives you some ideas about what should happen next:

- **Increase security (OWASP Top 10, TLS)**
  For real world usage TLS must be enforced. Also known common security issues should be checked e.g. OWASP Top 10
- **Increase usability (more AJAX)**
  AJAX is used already, but not everywhere. For the future we would like to add more for a better user experience.
- **Add test cases with high code coverage**
  Test driven development is important. We want to add good test cases with a high code coverage to be sure that code changes don't affect the application unintentionally.
- **Add more functionality**
  - Search for users -> useful for administrator
  - Mail notification service
    Users shall get notified when they are activated, accepted for an event etc.
  - SMS notification service
- **Create an android application for the employees**
  Most of the employees have smartphones. It would be nice to have an app with push notification so that the employees easily see when they have to work or when new jobs are available.
- **Check compatibility with different web browsers**
  We developed without taking care of other browsers than Mozilla Firefox and Google Chrome. Therefore we need to check the compatibility to different browsers e.g. Safari, Internet Explorer

Installation instructions

We used DigitalOcean for the deployment of our Django application. We used the Djangon on Ubuntu 14.04 droplet. This comes with an ubuntu image where a lot of configuration work is already done.
Our ".me" domain was part of the github.com education package. We got the free domain from nc.me. There we also had to change the A record of the DNSserver to point to our DigitalOcean server.

When you connect via SSH to the server it displays the credentials to upload files via SFTP.

Upload the BlueHive folder to the **/home/django/django_project/** directory. The SQLite database **db.sqlite3** needs to be uploaded into the same directory.

After this you need to change the global config file of the web application.
Modify the file **/home/django/django_project/django_project/settings.py**.

```
"""
Django settings for django_project project.

For more information on this file, see
https://docs.djangoproject.com/en/1.6/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.6/ref/settings/
"""

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
import os
BASE_DIR = os.path.dirname(os.path.dirname(__file__))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.6/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'lVpZhoskC3F8v7Y0c2UEhFS4arHqJ18P6q9K5fJDcU7iQiqovi'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

TEMPLATE_DEBUG = True
```

```
ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = (
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'BlueHive',
        'smartfields',
        'datetimewidget',
        'tastypie',
)

MIDDLEWARE_CLASSES = (
        'django.contrib.sessions.middleware.SessionMiddleware',
        'django.middleware.common.CommonMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
        'django.contrib.auth.middleware.AuthenticationMiddleware',
        'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
        'django.contrib.messages.middleware.MessageMiddleware',
        'django.middleware.clickjacking.XFrameOptionsMiddleware',
        'django.middleware.security.SecurityMiddleware',
)

ROOT_URLCONF = 'django_project.urls'

WSGI_APPLICATION = 'django_project.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.6/ref/settings/#databases

#DATABASES = {
#       'default': {
#       'ENGINE': 'django.db.backends.postgresql_psycopg2',
#       'NAME': 'django',
#       'USER': 'django',
#       'PASSWORD': 'WV803Bvdbe',
#       'HOST': 'localhost',
#       'PORT': '',
#       }
```

```
#}

DATABASES = {
        'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        }
}

# Internationalization
# https://docs.djangoproject.com/en/1.6/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.6/howto/static-files/

STATIC_URL = '/static/'
STATIC_ROOT = './BlueHive/static/BlueHive'
AUTH_USER_MODEL = 'BlueHive.CustomUser'
LOGIN_URL='/'
LOGIN_REDIRECT_URL='/'

MEDIA_ROOT = '.'
MEDIA_URL='/media/'
```

Also the file **/home/django/django_project/django_project/urls.py** must be changed so that all URLs of our application will be served.

```
...
urlpatterns = patterns('',
        url(r'^admin/', include(admin.site.urls)),
        url(r'^', include('BlueHive.urls', namespace="BlueHive")),

)
```

As you see in the INSTALLED_APPS part of the configuration file there have been additional apps installed. We need to install them now.

```
# pip install Django==1.8.6
# pip install django-smartfields
# pip install django-datetime-widget
# pip install django-tastypie
# aptitude install libpq-dev python-dev libjpeg-dev libpng-dev
# pip install -I pillow
```

Now only the NGINX web server which acts as something like a proxy needs to be configured correctly.
The file **/etc/nginx/sites-enabled/django**

```
upstream app_server {
	server 127.0.0.1:9000 fail_timeout=0;
}

server {
	listen 80 default_server;
	listen [::]:80 default_server ipv6only=on;

	root /usr/share/nginx/html;
	index index.html index.htm;

	client_max_body_size 4G;
	server_name _;

	keepalive_timeout 5;

	# Your Django project's media files - amend as required
	location /media  {
	#alias /home/django/django_project/BlueHive/static;
	alias /home/django/;
	}

	# your Django project's static files - amend as required
	location /static {
	alias /home/django/django_project/BlueHive/static;
	}

	# Proxy the static assests for the Django Admin panel
```

```
        location /static/admin {
        alias /usr/lib/python2.7/dist-packages/django/contrib/admin/static/admin/;
        }

        location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_pass http://app_server;
        }
}
```

Now you need to restart the gunicorn service and after this the whole server:

```
# service gunicorn restart
# reboot
```

Now the server should be up and running.
If there are some problems you can check specially the gunicorn logs:

 # tail -30 /var/log/upstart/gunicorn.log
or start the django internal webserver for testing:
root@BlueHive2:/home/django/django_project# **python manage.py runserver**

References

- https://docs.djangoproject.com/en/1.8/
- http://tastypieapi.org/
- https://github.com/asaglimbeni/django-datetime-widget
- http://agiliq.com/blog/2015/03/getting-started-with-django-tastypie/
- http://stackoverflow.com/questions/14671336/how-to-use-api-key-authentication-by-django-tastypie
- https://django-tastypie.readthedocs.org/en/latest/authentication.html
- http://stackoverflow.com/
- https://www.youtube.com/watch?v=3Dgfp6hLznA
- http://www.dropzonejs.com/