

# Application Development for Mobile and Ubiquitous Computing

## Seminar Task Second Presentation

Group №: 20

Team: Anton Caceres  
Andrii Chaichenko

# Why mobile advertising application?

---

- In Germany people are generous
  - they give lots of good things 'zu verschenken'
  - usually just throwing them to the street container
- Because to post or get an offer you would usually need:
  - computer for web brosing
  - phone to call owner
  - maybe camera to take photos
- However, you can do it all-at-once with our mobile app:
  - no more need to search
  - inbox based on feeds
  - search terms, location-based search
  - mail, browsing, maps and calls all in one place

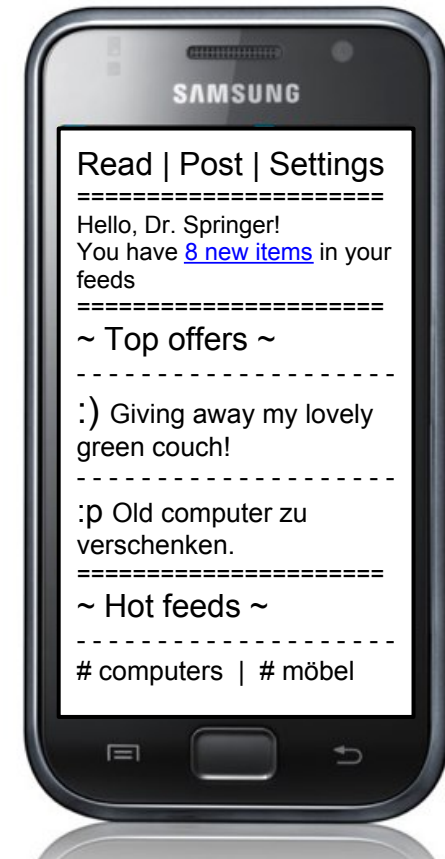
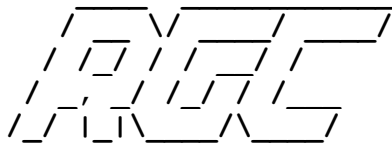
# Application Scenario

---

- user can specify search terms for products, as well as perform standard search
- based on given search terms user receives offers
- offers arrive in feed-like view
- direct contact with seller
- possibility to filter by location (i.e. in 5 km radius)
- post your own ad
- add a picture to your post from a phone camera

# Basic features and interface

- Key features - users can read feeds and post new items
- Feeds are grouped into slidable lists of offers
- Everything else is in "settings" tab.
- Mockup inspired by ASCII Art



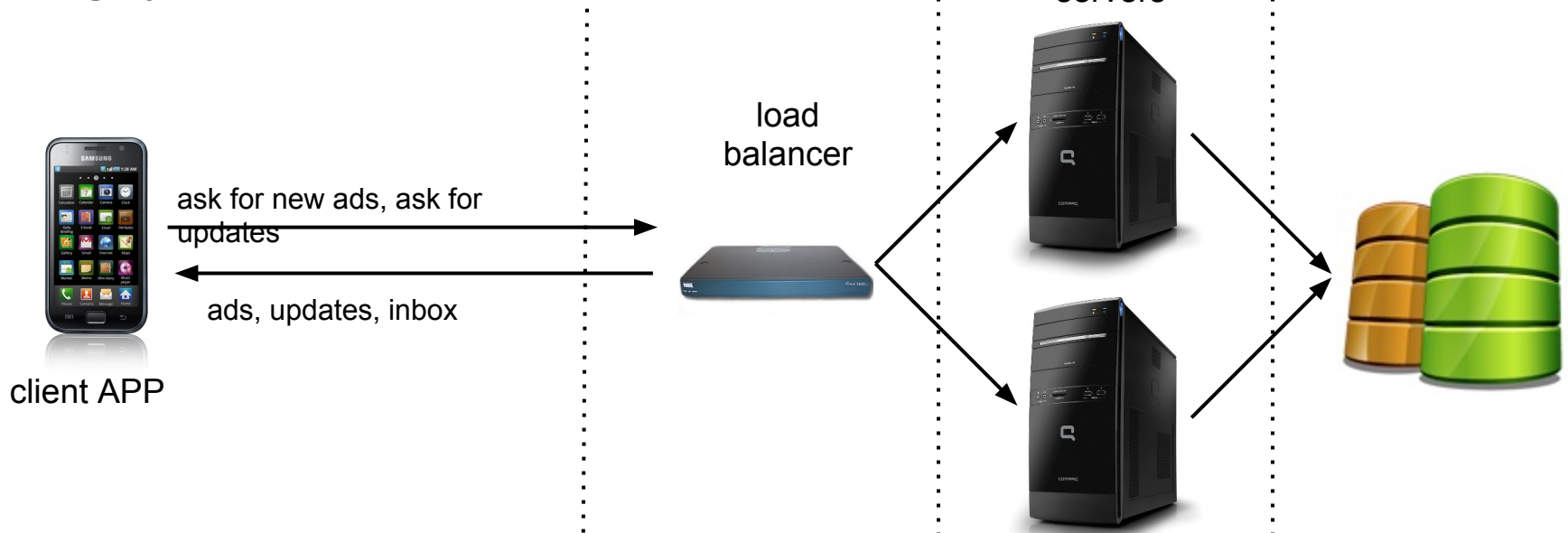
# Basic features and interface

- Content is delivered to user (concept of RSS feed), no need to search in different places
- User needs just 1 device to arrange a meeting (search, email, call, lookup the map)
- In case we have enough time, user would also be able to post offers from the single device.
- Multi-platform!



# Technologies

- Nginx - server load balancing
- Django/Python - server logic
- Python - crawling module
- Postgre - database
- PhoneGap - client middleware
- HTML5 & Javascript & jQuery mobile - client app
- Android 2.3 smartphone
- lightweight client\server communication via JSON
- RESTful API



# Project diagram

## django.contrib.auth

Message	
id	AutoField
user	ForeignKey
message	TextField

user (\_message\_set)

User	
id	AutoField
username	CharField
first_name	CharField
last_name	CharField
email	EmailField
password	CharField
is_staff	BooleanField
is_active	BooleanField
is_superuser	BooleanField
last_login	DateTimeField
date_joined	DateTimeField
groups	ManyToManyField
user_permissions	ManyToManyField

django\_user (profile)

groups (user)

Group	
id	AutoField
name	CharField
permissions	ManyToManyField

user\_permissions (user)

permissions (group)

Permission	
id	AutoField
name	CharField
content_type	ForeignKey
codename	CharField

## core

Offer	
id	AutoField
text	TextField
timestamp	DateTimeField
rating	FloatField
author	ForeignKey
category	IntegerField
location	ForeignKey
reward	ForeignKey
status	IntegerField
email	EmailField
phone	CharField
conversation	TextField

author (offer)

reward (offer)

Reward	
title	CharField
slug	SlugField
image	CharField

offers (feeds)

location (offer)

UserProfile	
id	AutoField
django_user	ForeignKey
display_name	CharField
phone	CharField
location	ForeignKey
feed	ForeignKey
keywords	ManyToManyField

keywords (users)

feed (userprofile)

location (userprofile)

Keyword	
id	AutoField
keyword	CharField

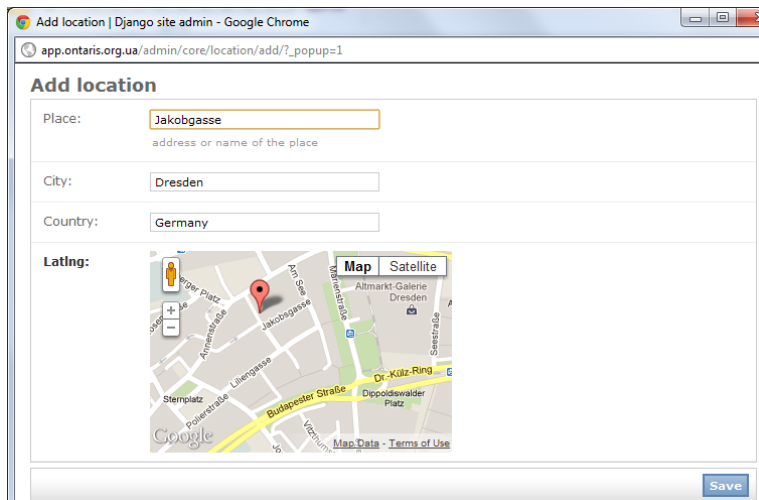
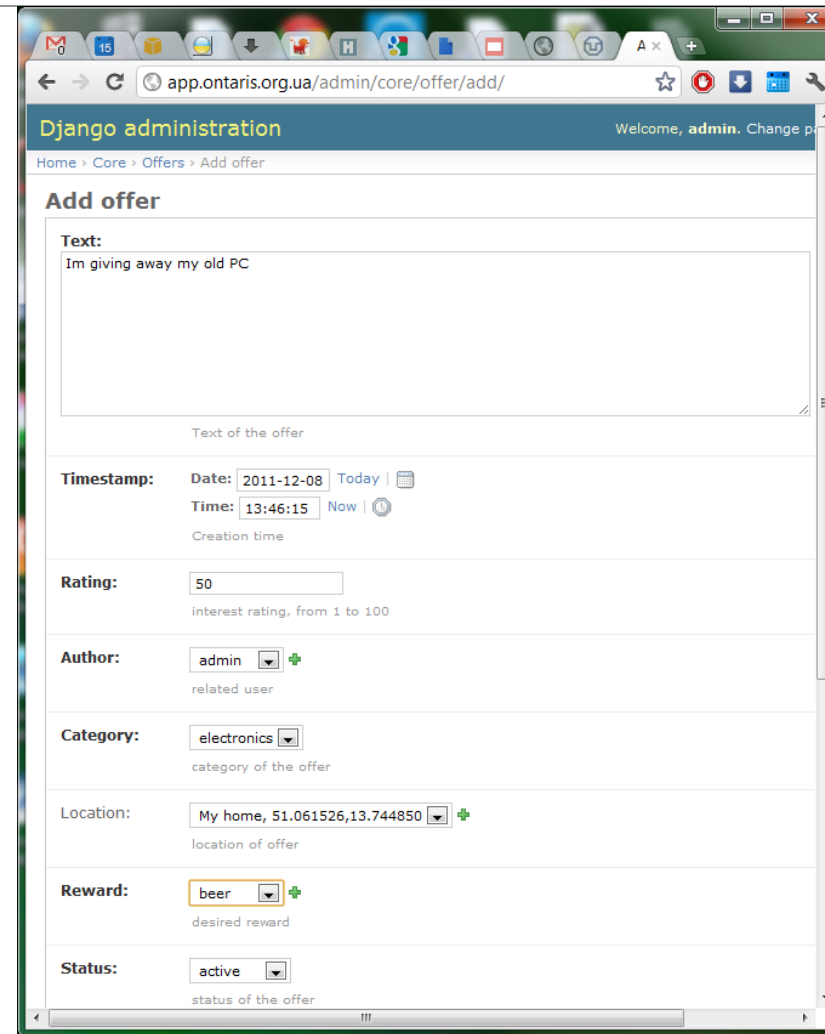
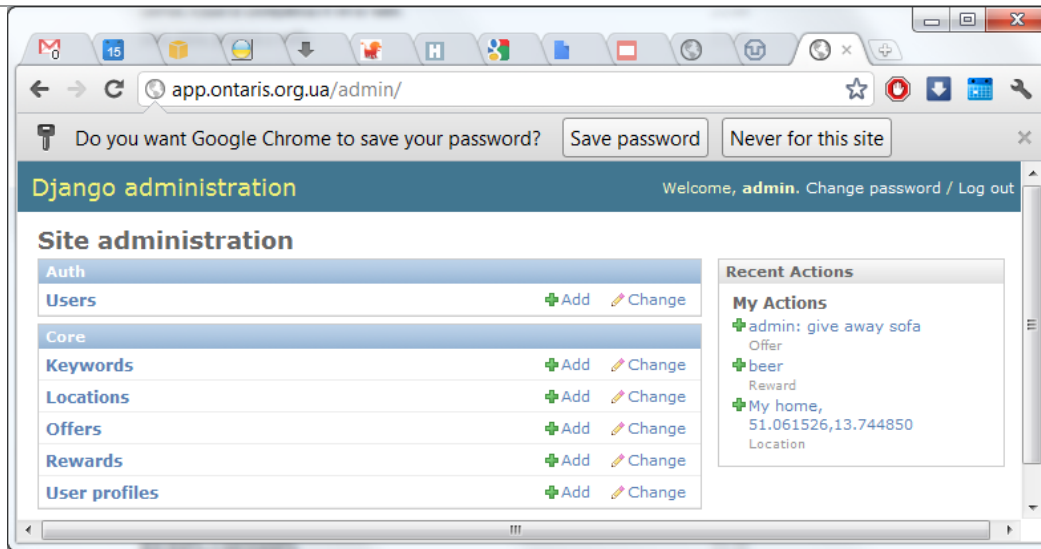
Feed	
id	AutoField
offers	ManyToManyField

Location	
id	AutoField
place	CharField
city	CharField
country	CharField
latlng	CharField

## django.contrib.sessions

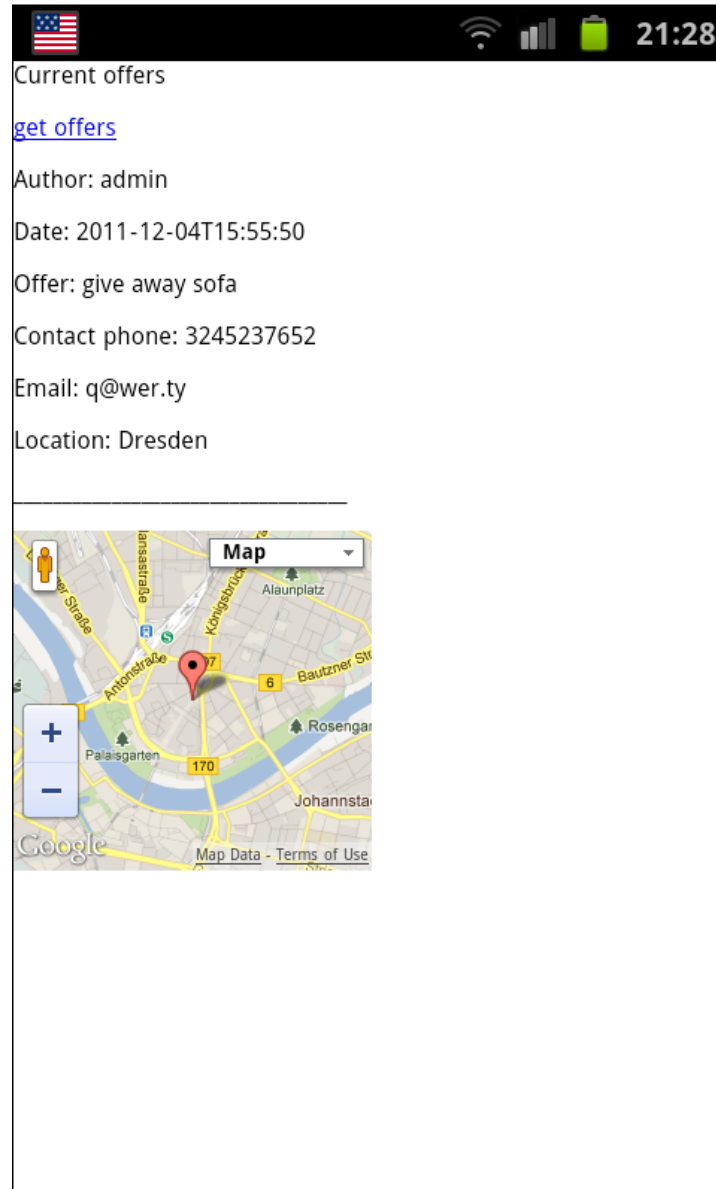
Session	
session_key	CharField
session_data	TextField
expire_date	DateTimeField

# Screenshots







# Screenshots





- Crossplatform mobile application development
- Server load balancing
- Secure data transfer and authentication
- Work under unstable network connection
- Real-time updates
- Suitable for low bandwidth

-  Conceptional design
  - Define use cases
  - Decide on platforms and tools
  - Freeze feature-list

 done  
 next

-  Set up backend:
  - DB server
  - Django application server
  - front-end NginX server

-  Create an external crawling daemon
  - Constantly running Python daemon
  - gets new posts from ebay, cybersax etc
  - Pushes results via RESTful API to main server

-  Write application logic
  - Design models
  - Code controllers
  - Add middleware (e.g. authentication, http sessions)
  - Hook up RESTful API

- ➔ ■ Create client application
  - Code controllers
  - Code html views
  - Debug damn Javascript
  - Test the whole thing

➔ ■ Profit

# The Most Important Slide

---

Questions?