Value-Added Services for Local Search: Innovative Recommender Systems and Routing Models

F. Ricci, M. Braunhofer, M. Reinstadler, M. Elahi et al.

Free University of Bozen - Bolzano
Faculty of Computer Science
fricci@unibz.it

unibz

TIS innovation park

SASA
SOCIETÀ AUTOBUS SERVIZI D'AREA
STÄDTISCHER AUTOBUS SERVICE AG

Landesverband der
Tourismusorganisationen
Südtirols

Comune di Bolzano
Stadtgemeinde Bozen

Outline

- STS Place Recommendation App
- Comino Routing Engine
- Parking Lot Occupancy Prediction

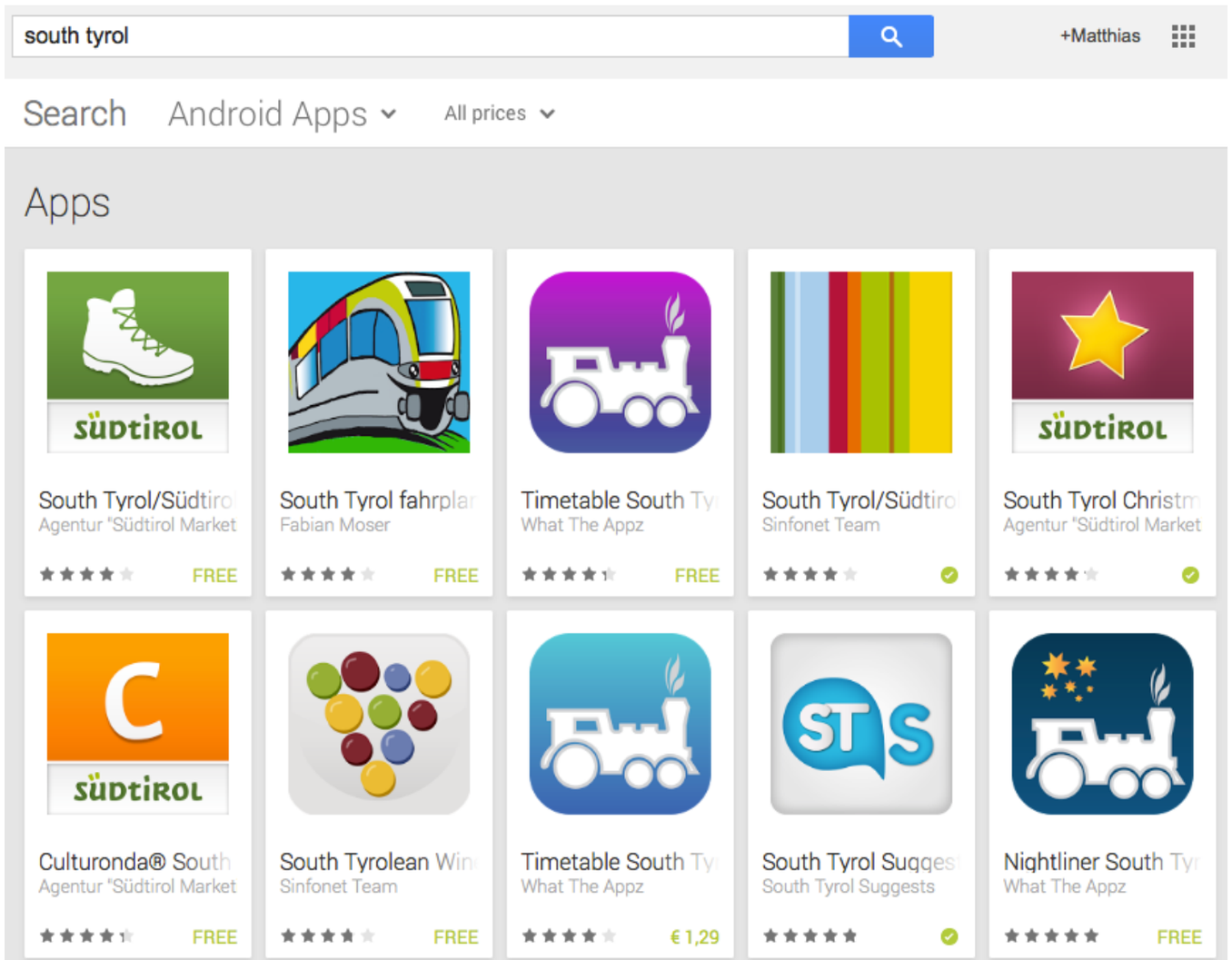
Outline

- STS Place Recommendation App
- Comino Routing Engine
- Parking Lot Occupancy Prediction

STS (South Tyrol Suggests)

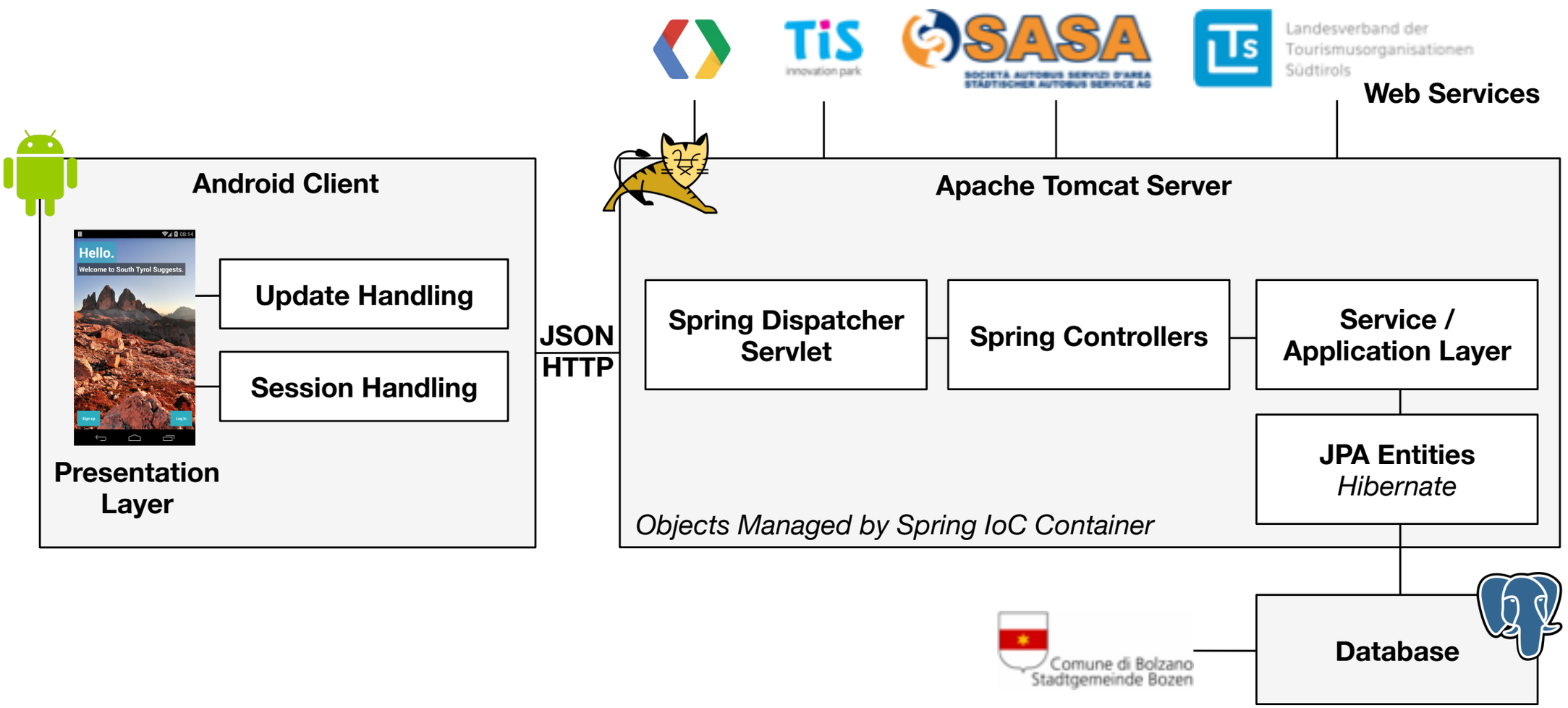
- Our **Android app** on Google Play that supports the following functionalities:
  - Intelligent recommendations for POIs in South Tyrol that are adapted to the current contextual situation of the user (e.g., weather, location, parking status)
  - Eco-friendly routing to selected POIs by public or private transportation means
  - Search for various types of POIs across different data sources (i.e., LTS, Municipality of Bolzano)
  - User personality questionnaire for preference elicitation support

Statistics



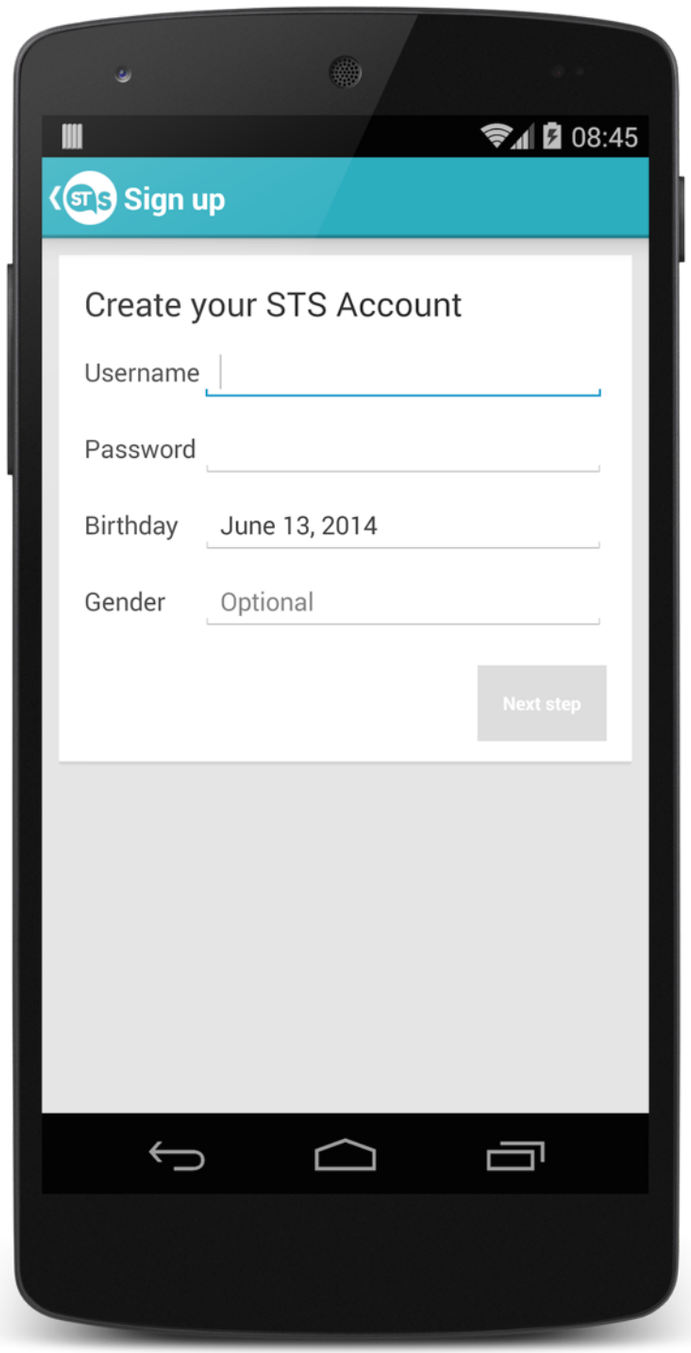
- App usually shown in the **top-10 search results**
- Current/total installs: **164 / 575**
- Avg. rating/total #: **4.92 / 13**

Software Architecture and Implementation

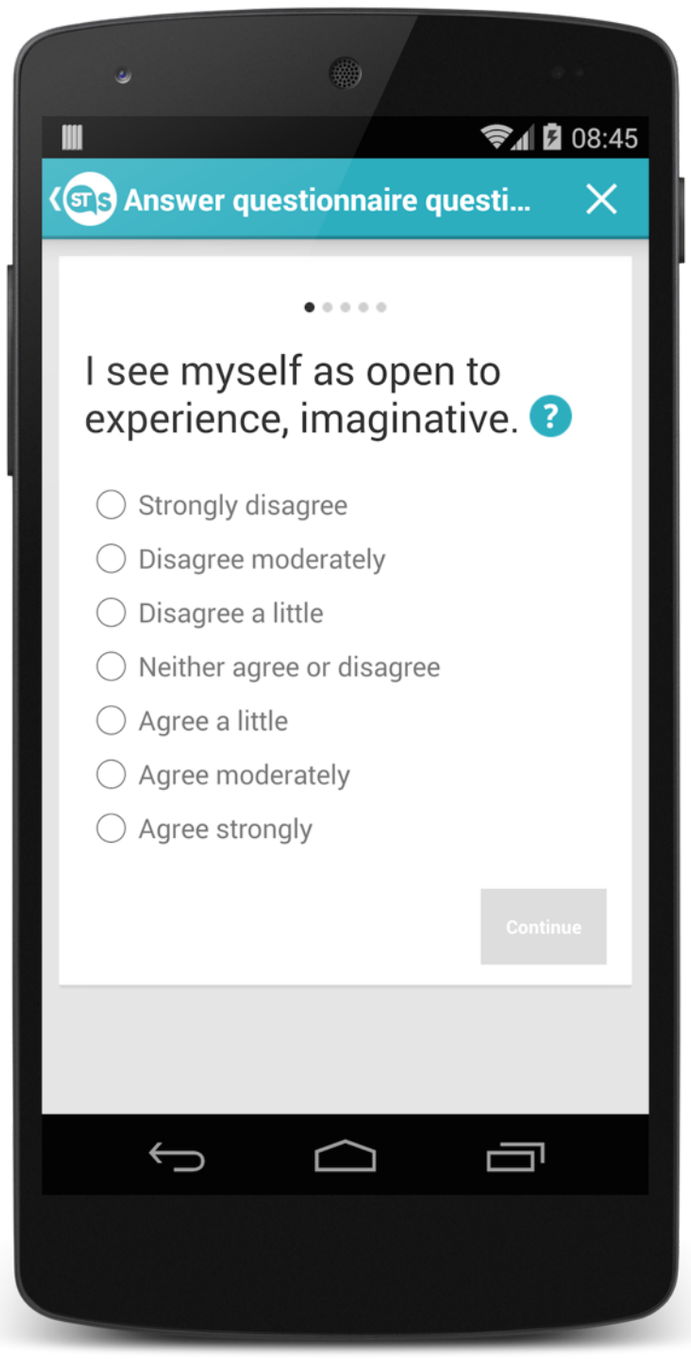


Interaction with the System

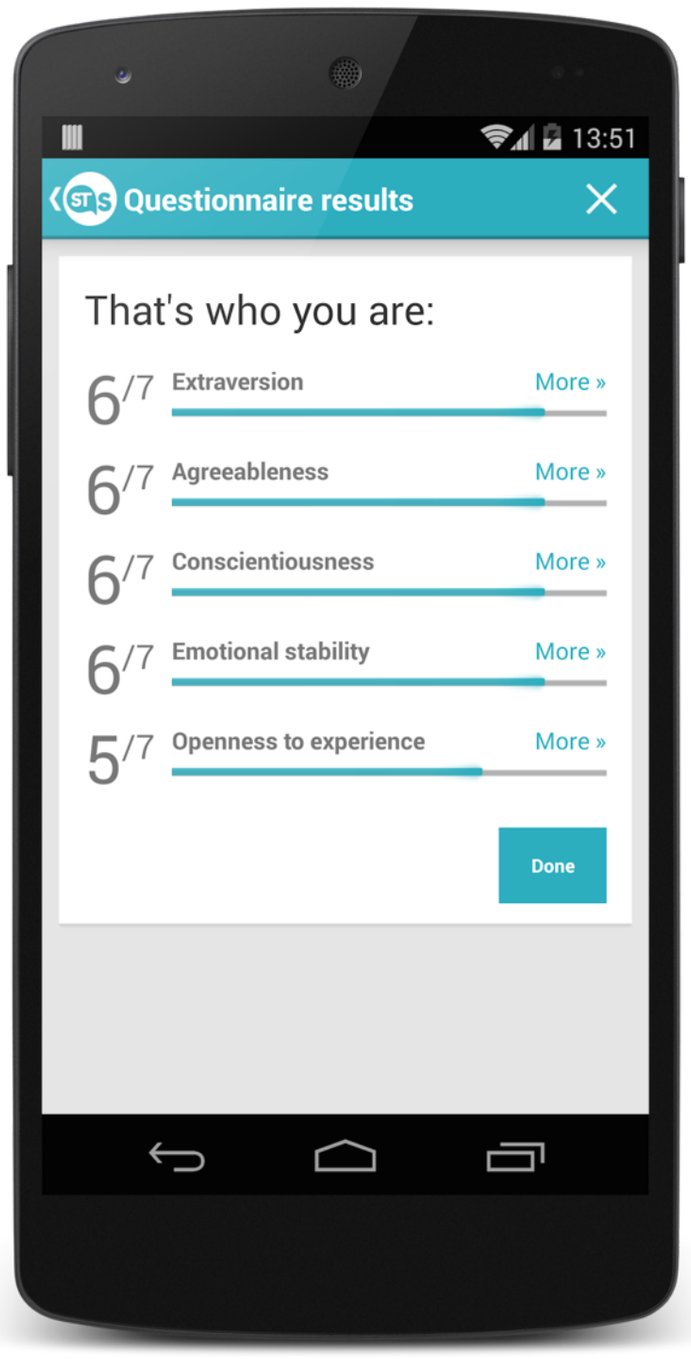
Interaction with the System



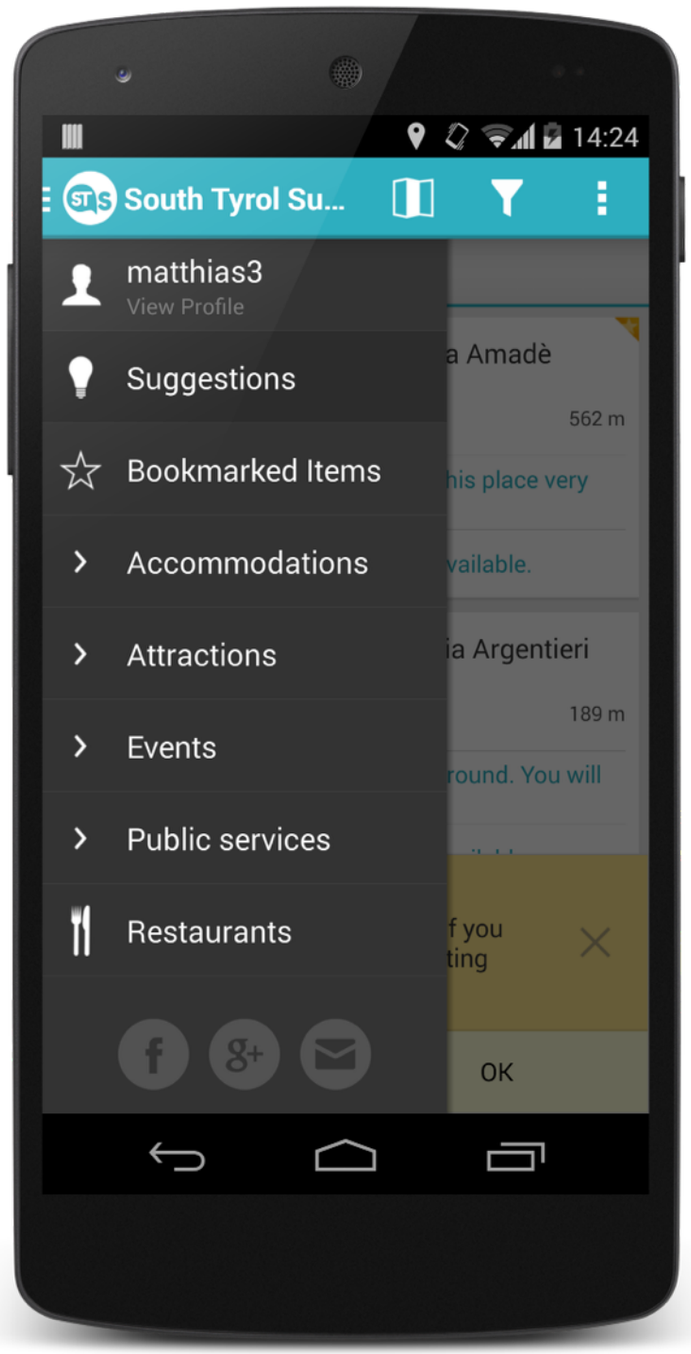
Interaction with the System



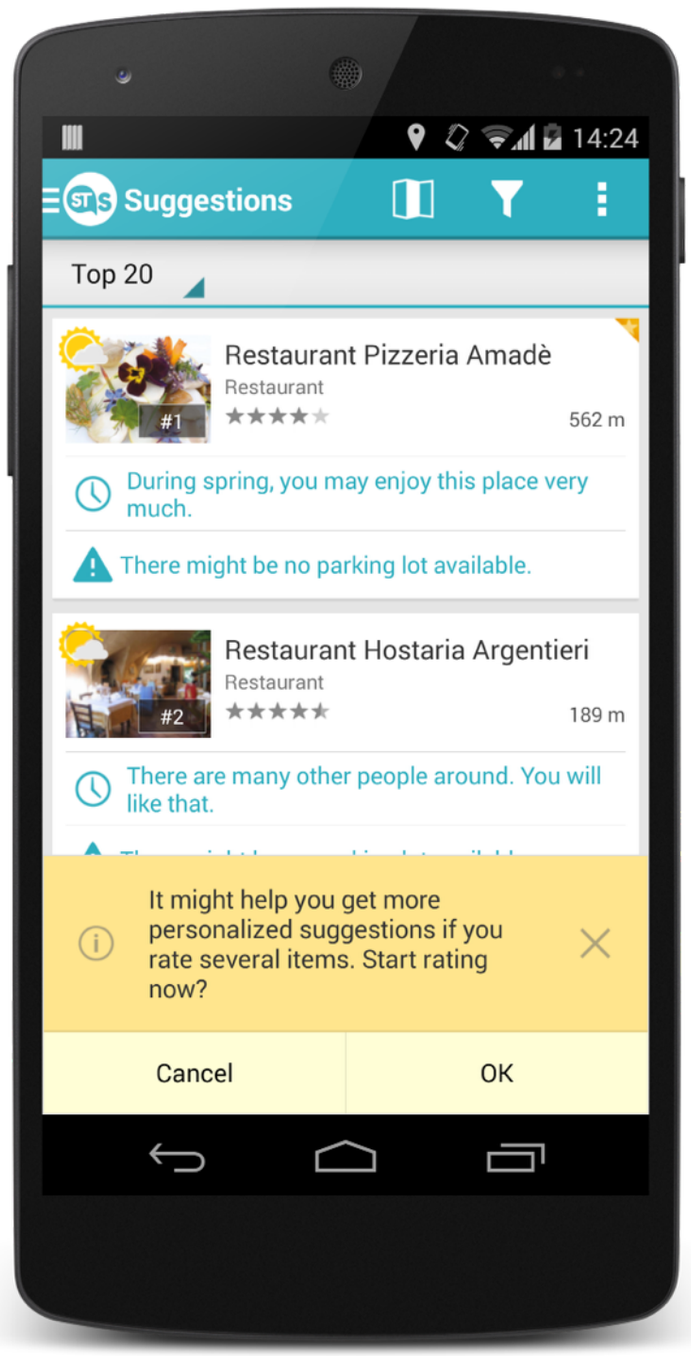
Interaction with the System



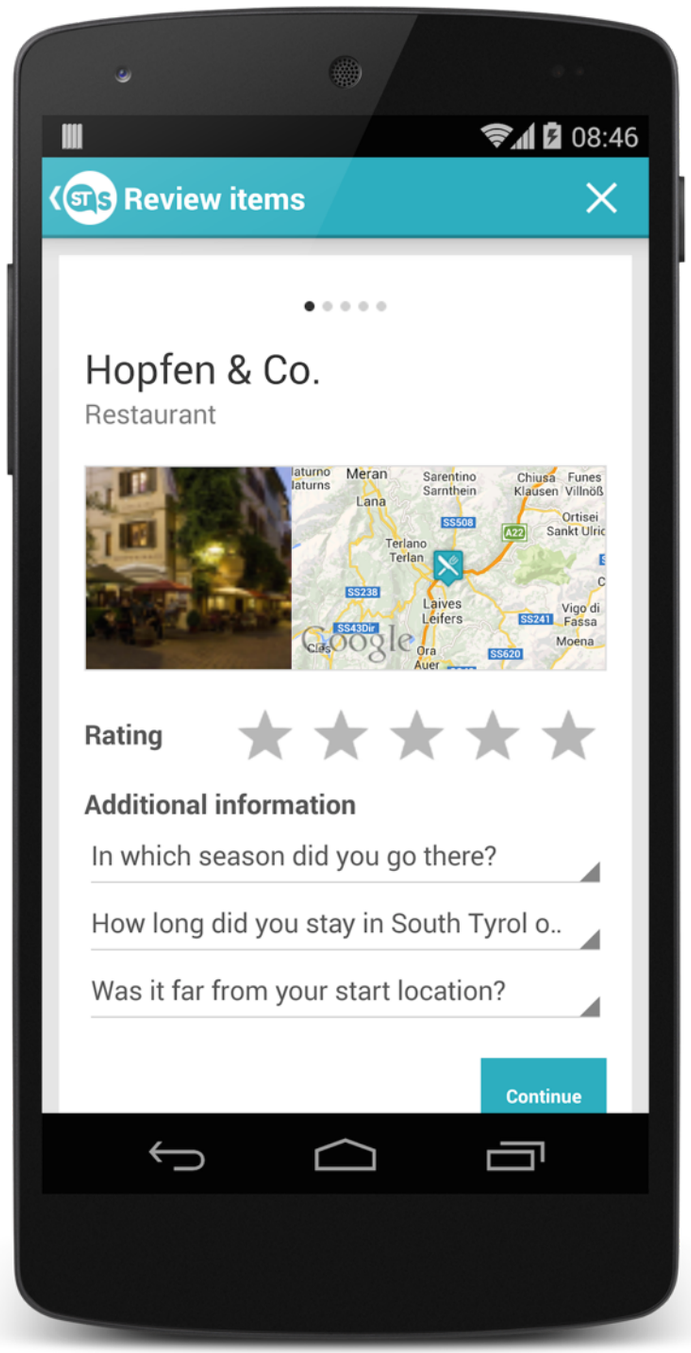
Interaction with the System



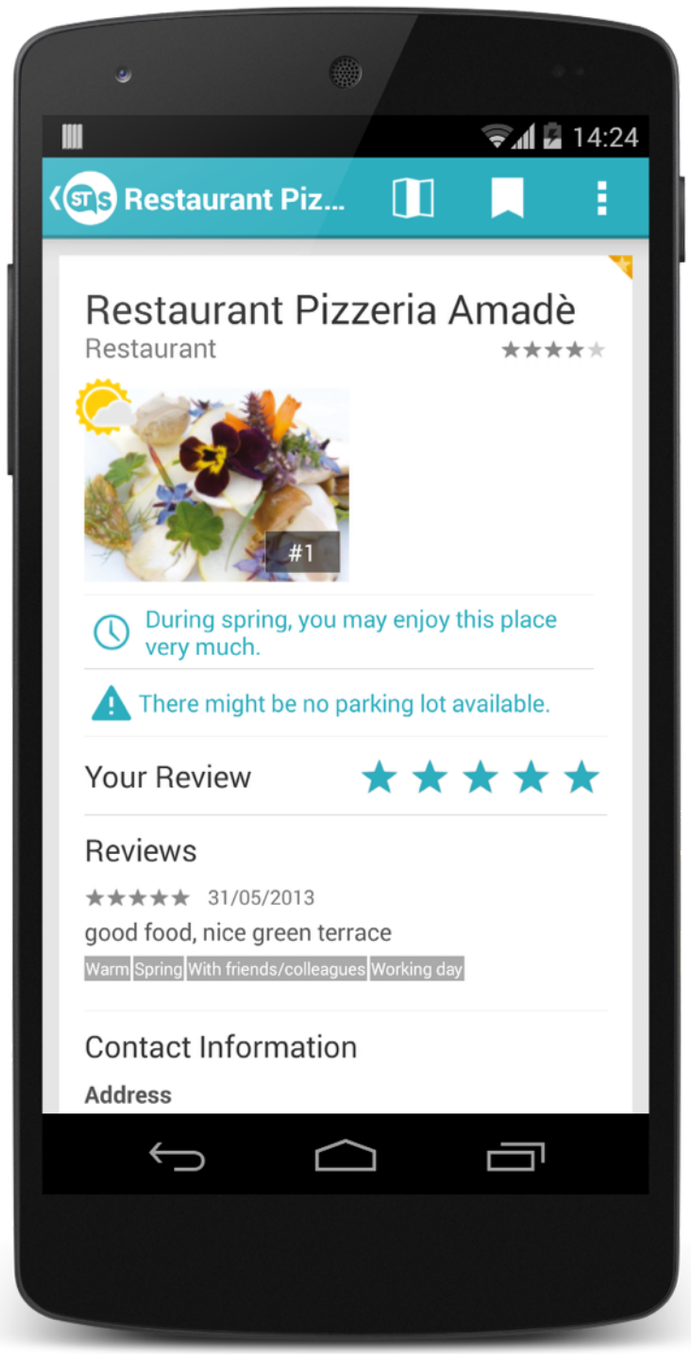
Interaction with the System



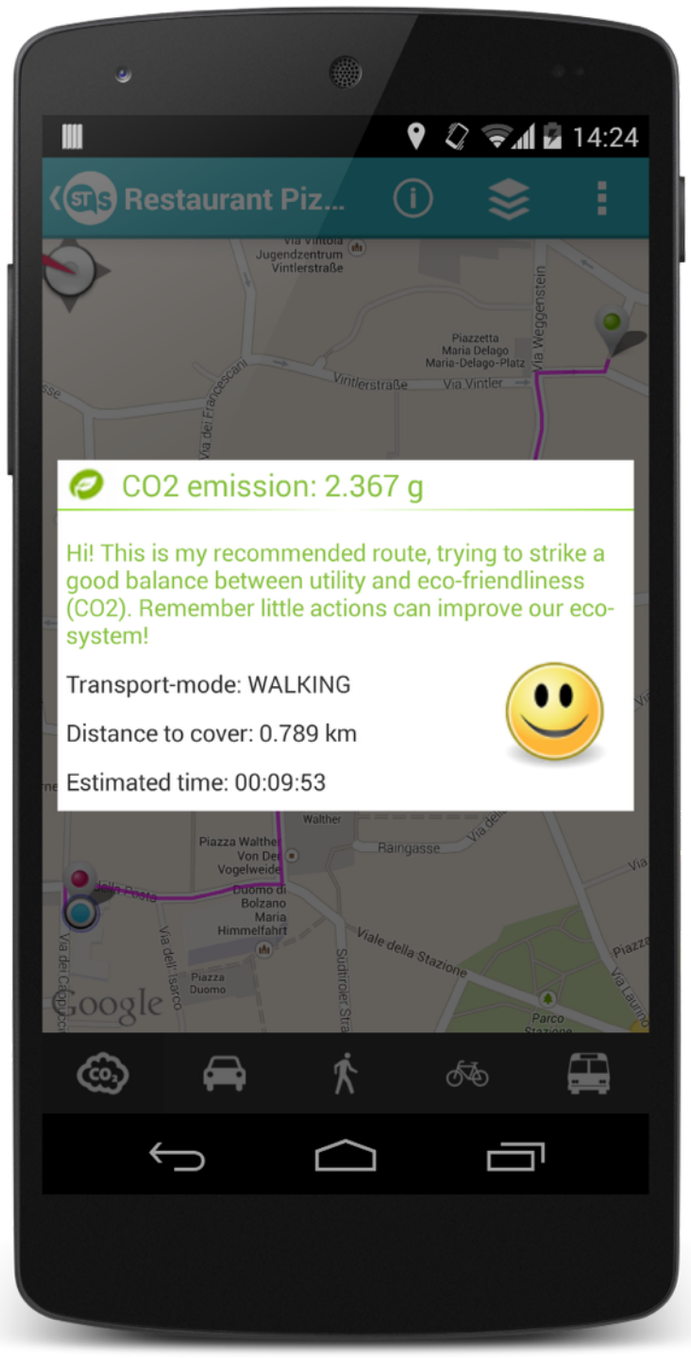
Interaction with the System



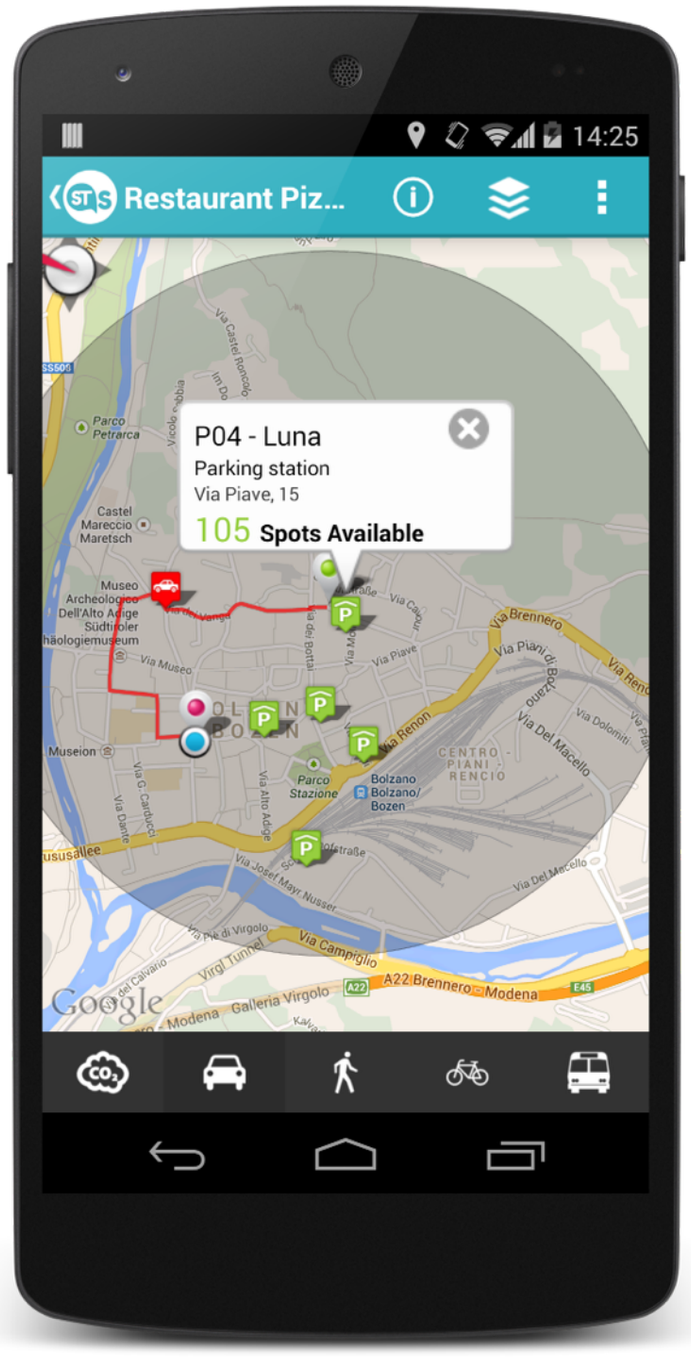
Interaction with the System



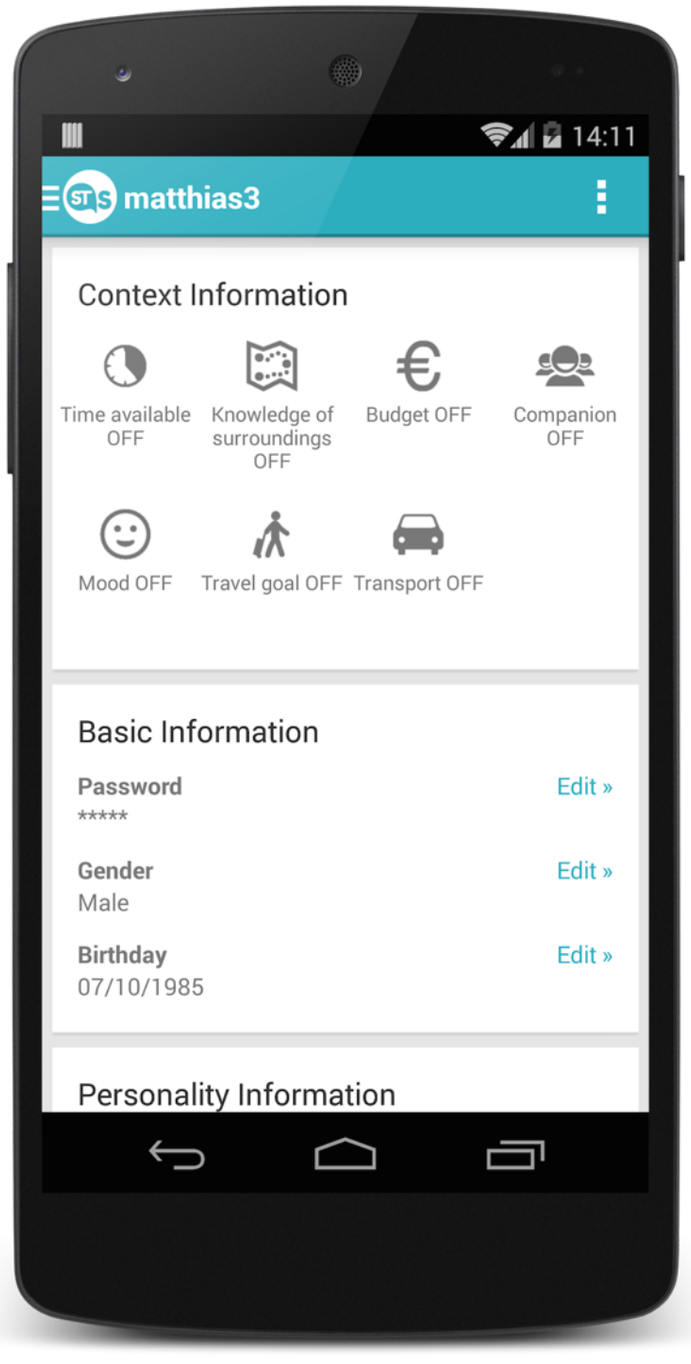
Interaction with the System



Interaction with the System



Interaction with the System



Recommendation Task

- Core computations of recommender systems:
  - **Collection of user preferences (ratings):** collect user feedback (ratings) on items to learn the user preferences
  - **Rating prediction:** a model must be built to predict ratings for items not currently rated by the user
  - **Item selection:** a model must be built that selects the N most relevant items for the user

Rating Prediction Algorithm (1/2)

- Rating prediction algorithm is based on **Matrix Factorization (MF)**
  - **Basic idea of MF:** predict unknown ratings by discovering some latent features that determine how a user rates an item; features associated with the user should match with the features associated with the item

r
5 x 4 matrix

r_{11}	r_{12}	r_{13}	r_{14}
r_{21}	r_{22}	r_{23}	r_{24}
r_{31}	r_{32}	r_{33}	r_{34}
r_{41}	**r_{42}**	r_{43}	r_{44}
r_{51}	r_{52}	r_{53}	r_{54}

=

q
5 x 3 matrix

a	**b**	**c**

p
3 x 4 matrix

	x		
	y		
	z		

$$r_{42} = (a, b, c) \cdot (x, y, z) = a * x + b * y + c * z$$

Rating prediction

$$\hat{r}_{ui} = q_i^T p_u$$

User preference factor vector

Item preference factor vector

Rating Prediction Algorithm (2/2)

- **Context-Aware Matrix Factorization (CAMF):** extends standard MF by incorporating baseline parameters for each contextual condition and item pair to capture the deviation of the rating for an item produced by the contextual conditions

$$\hat{r}_{uic_1,\ldots,c_k} = q_i^T (p_u + \sum_{a \in A(u)} y_a) + \bar{i} + b_u + \sum_{j=1}^{k} b_{ic_j}$$

Captures the rating deviation due to context (e.g., weather, parking)

Rating = 4

Item average	User bias	Context bias	Preference factor (user-demographics-item-interaction)

c_j	contextual condition j
q_i	latent factor vector of item i
p_u	latent factor vector of user u
A(u)	set of user attributes
y_a	latent factor vector of user attribute a
$\bar{i}$	average rating of item i
b_u	baseline for user u
b_{ic_j}	baseline for item-contextual condition ic_j

Evaluation

- **Several user studies** involving > 100 test users
  - Test users were students, colleagues, or other people recruited at the Klimamobility Fair and Innovation Festival
- **Obtained results:**
  - Recommendation model successfully exploits the weather conditions at POIs and leads to a higher user's perceived recommendation quality and choice satisfaction
  - Implemented active learning strategy increases the number of acquired ratings and recommendation accuracy
  - Users largely accept to follow the supported human-computer interaction and find the user interface clear, user-friendly and easy to use

Next Steps

- Integration of the **Comino routing engine**
- Allow users to plan **future visits** to POIs
- Provide users with **push recommendations**
- Exploit **activity and emotion information** inferred from wearable devices in the recommendation process

Outline

- STS Place Recommendation App
- Comino Routing Engine
- Parking Lot Occupancy Prediction

Comino

- **Multimodal routing engine** that combines several transportation networks
  - Walking
  - Public transport system
  - Car
  - Bicycle
- Shall be used as a service by the STS and SASAbus apps

Comino API

- **Beta version** available at http://comino.inf.unibz.it
  - Computes point-to-point shortest path for a given arrival/departure time
  - Simple GUI
  - RESTful API in JSON
- **Limitations:**
  - Bus schedules must be updated
  - Not scalable for large networks and massive parallel requests

Next Steps

- Access and integrate **current bus schedules**
- **Improve performance** to efficiently serve many parallel requests for large networks
- **Support additional modalities:** bicycle, car
- **Provide additional functionalities:** top-k shortest routes, limit the number of changes between modalities, maximum time for walking (e.g., elderly people)

Outline

- STS Place Recommendation App
- Comino Routing Engine
- Parking Lot Occupancy Prediction

Parking Lot Occupancy Prediction

- **Information about parking lot occupancy in the near future can be very useful**
  - Plan a journey appropriately
  - Save time when looking for a parking lot
  - Reduce the traffic in a city

Occupancy Prediction: Architecture (1/2)

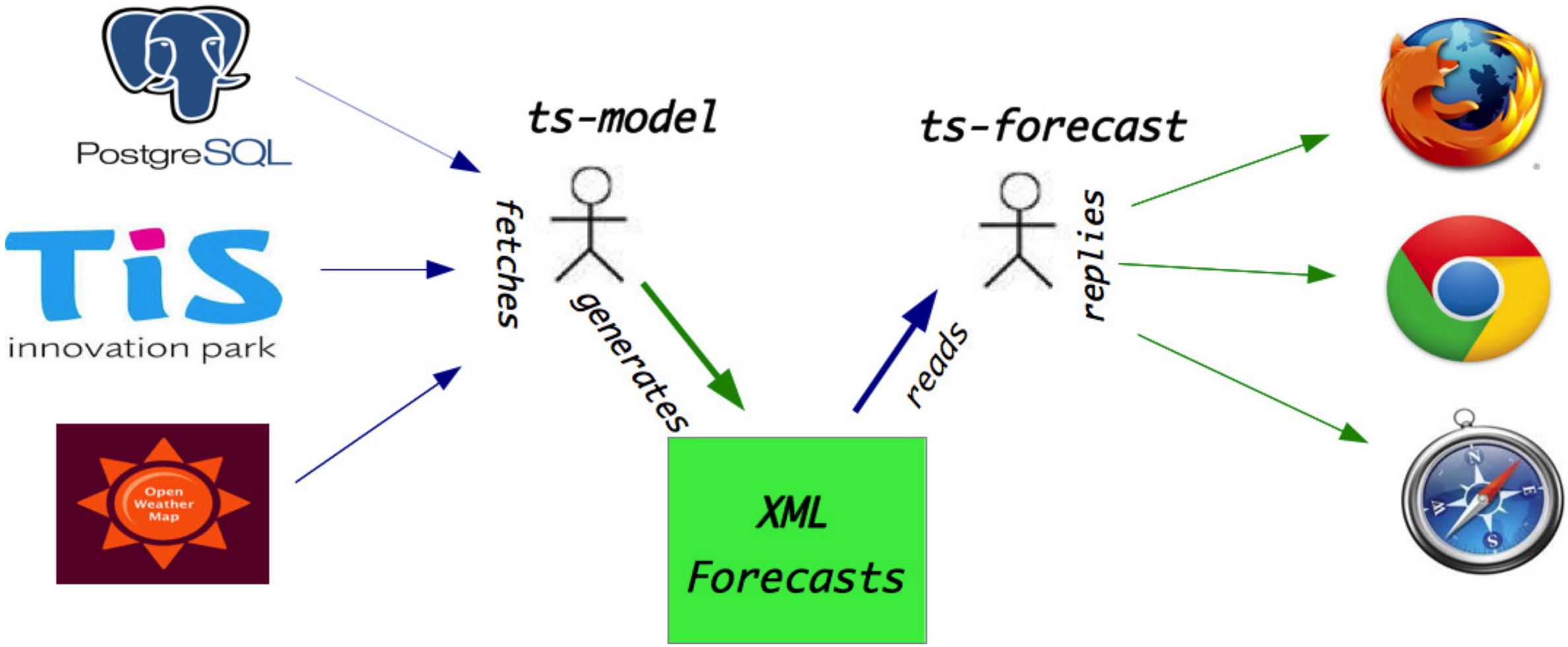
1. Collect historical information (parking lots, weather, holidays, etc.)

2. Build models for estimating parking occupancy

3. Generate and store predictions

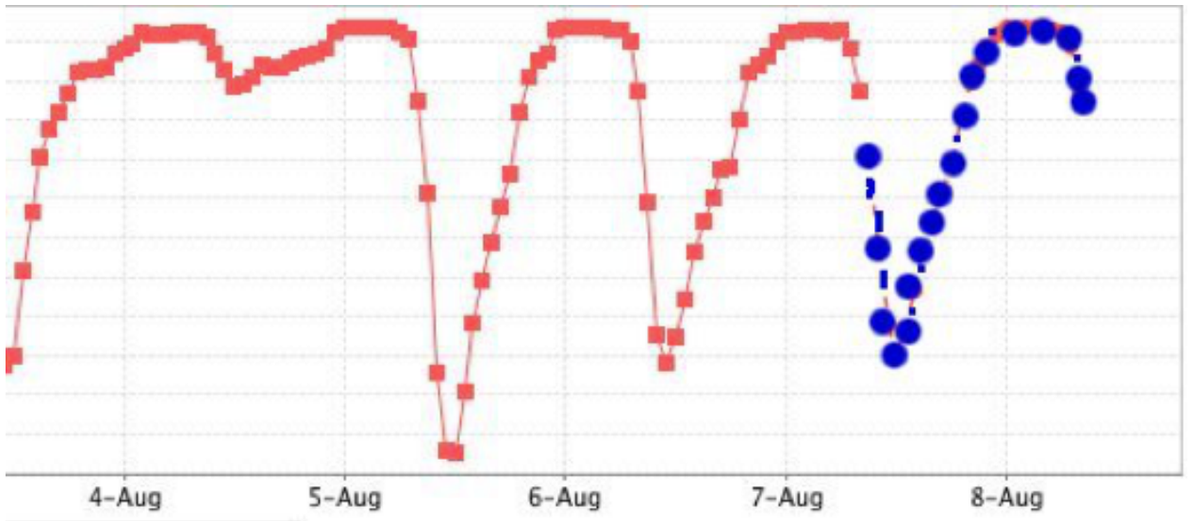
4. Reply to web service requests

Occupancy Prediction: Architecture (2/2)

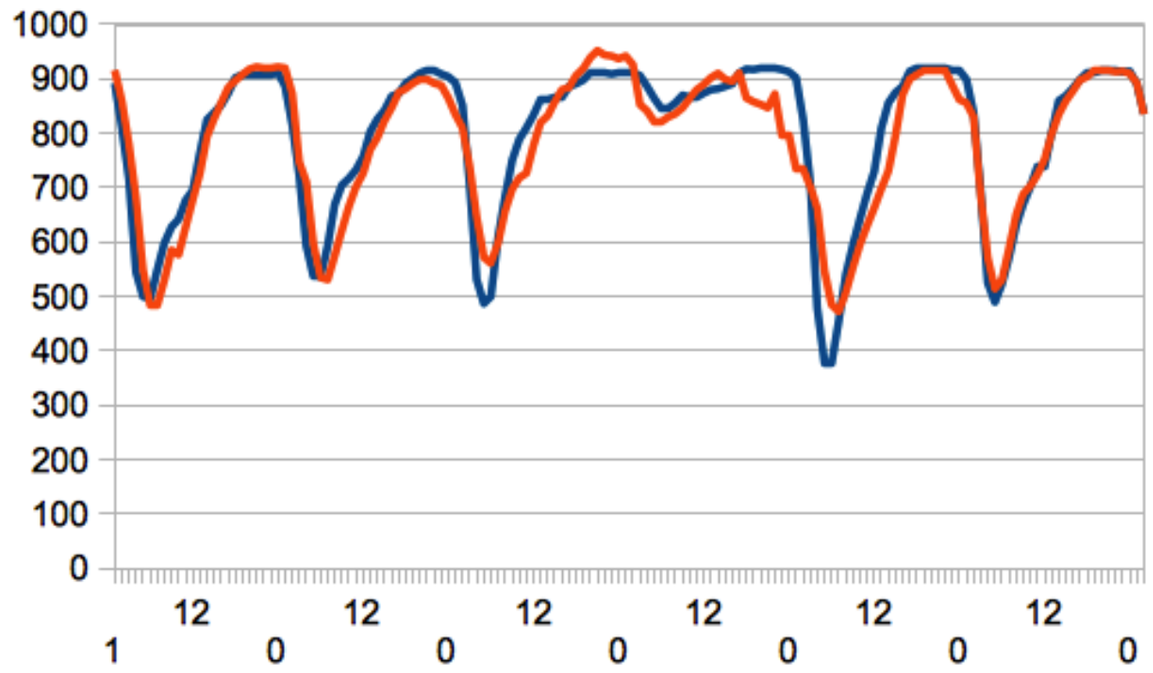


Occupancy Prediction: Example

- Parking lot 108
  - Observed free slots
  - 24 hours prediction



- Comparison
  - Real values (red)
  - Predicted values (blue)



Next Steps

- Evaluate and improve the forecasting algorithm
- Exploit the forecasts in various tools and services
- Apply prediction approach to other fields (e.g., traffic prediction)