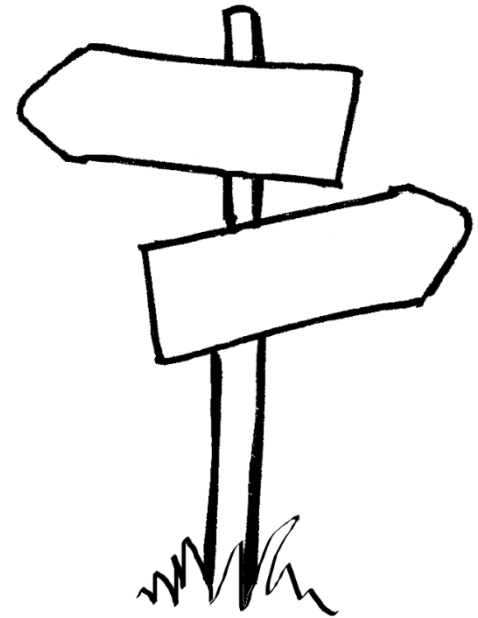




Using RSS Feeds to Support Second Language Acquisition

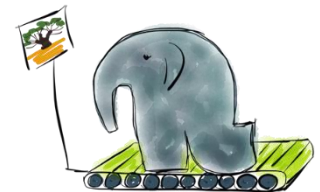
Roadmap

1. Introduction to Zeeguu
2. Demo
3. Architecture
4. Article Recommender
5. Conclusion

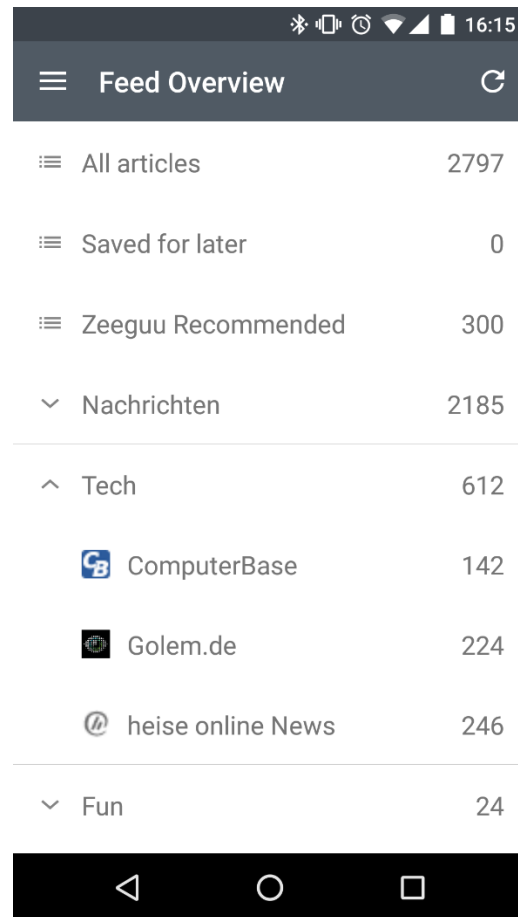


1. Introduction to Zeeguu




- Three fundamental principles
 - Only read the stuff you like
 - Have your words everywhere with you
 - Practice with personalized exercises
- Introducing Zeeguu Reader for Android
 - RSS Reader with Feedly synchronization
 - Learn anywhere while reading
 - Provides article recommendations



2. Demo



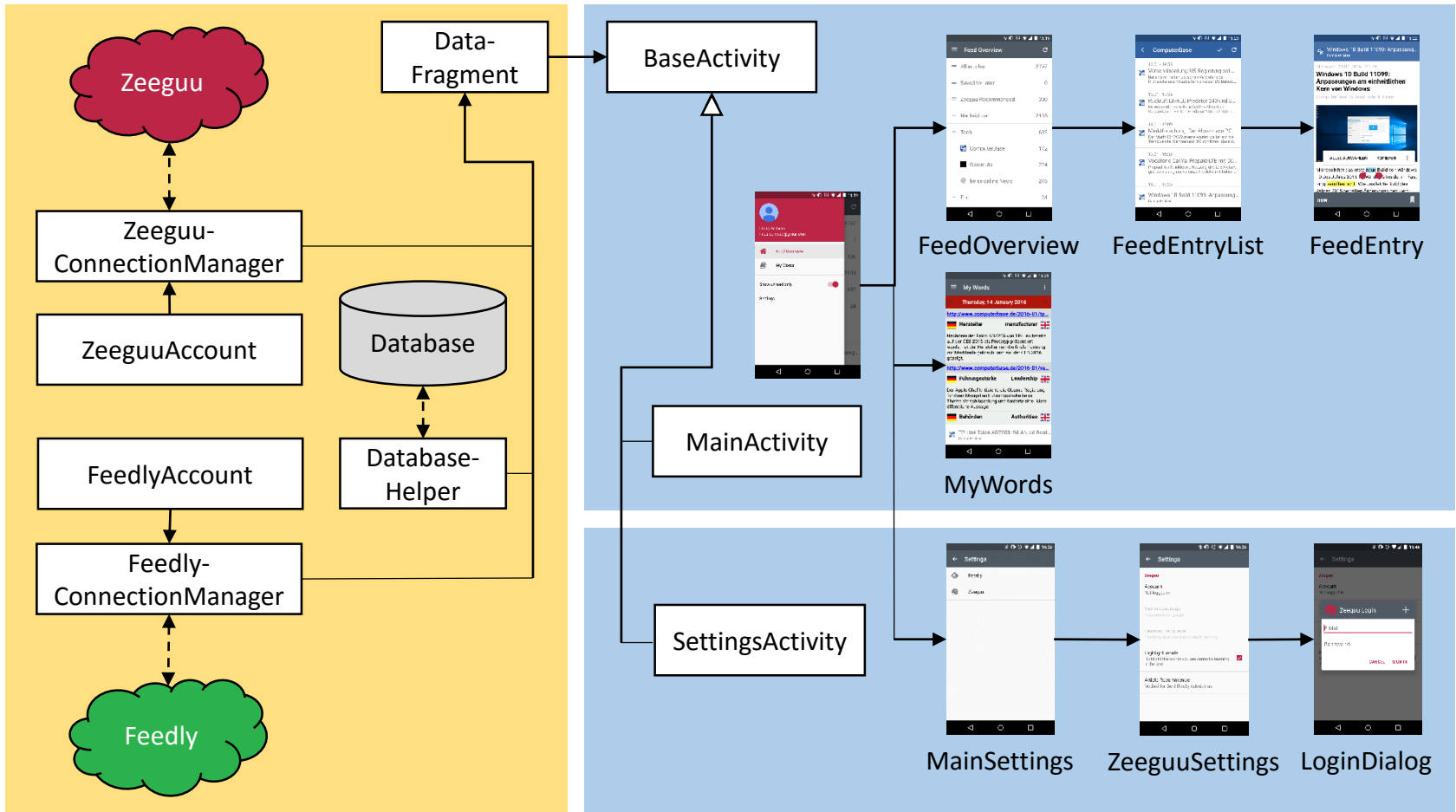
The screenshot shows a mobile application interface. At the top is a dark status bar with icons for Bluetooth, signal strength, alarm clock, Wi-Fi, and battery, along with the time 16:15. Below this is a dark header bar with a hamburger menu icon, the text 'Feed Overview', and a refresh icon. The main content area is a list of items, each with a list icon (three horizontal lines) and a count. The items are: 'All articles' (2797), 'Saved for later' (0), 'Zeeguu Recommended' (300), 'Nachrichten' (2185), 'Tech' (612), 'ComputerBase' (142), 'Golem.de' (224), 'heise online News' (246), and 'Fun' (24). The items are separated by horizontal lines. At the bottom is a black navigation bar with three white icons: a back arrow, a circle, and a square.

☰ All articles	2797
☰ Saved for later	0
☰ Zeeguu Recommended	300
▼ Nachrichten	2185
^ Tech	612
 ComputerBase	142
 Golem.de	224
 heise online News	246
▼ Fun	24

3. Architecture: User Interface

- Activity
 - Main application component
 - Provide the window for the user interface
 - Handle communication between fragments
 - This app: MainActivity, SettingsActivity
- Fragment
 - Reusable portion of user interface
 - Dynamically replaced by activity
 - This app: used whenever possible

3. Architecture: Overview

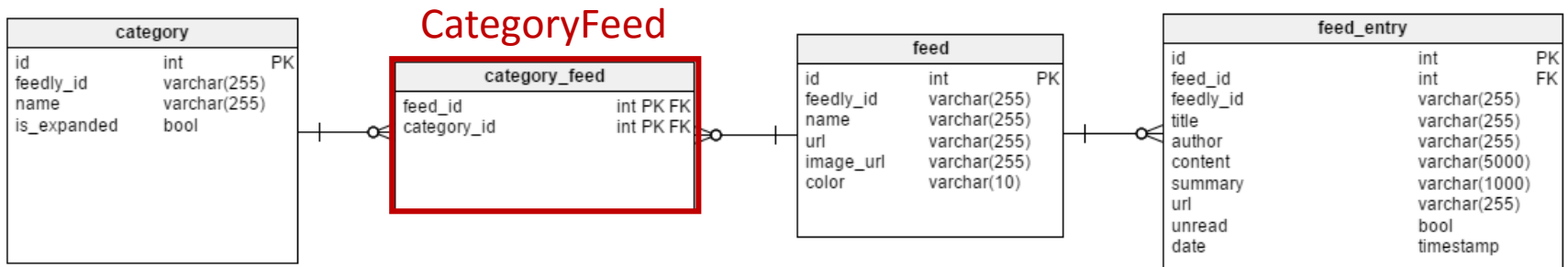


3. Architecture: Back End

- ConnectionManager
 - Classes to communicate with Zeeguu and Feedly API
 - Uses Volley
- Account
 - Manages user data
 - ZeeguuAccount
 - Stored in SharedPreferences
 - FeedlyAccount
 - Handles synchronization
 - Interface to Database

3. Architecture: Back End

- ORM (Object Relational Mapping)
 - Implemented with ORMLite
 - Works with annotations
 - Uses DAO pattern (Data Access Objects)
 - Flexible QueryBuilder to easily construct queries
 - Does not directly support many-to-many relations



4. Article Recommender

- Helps the user to find suitable articles to read
- Presented in “Zeeguu Recommended” category
- Implemented on the Zeeguu server
- Two components
 - Difficulty
 - Learnability

4. Article Recommender: Idea

- Analyzes text on word-based level
- Two metrics used to estimate difficulty
 - KnownWordProbability
 - RankedWord (Word frequency lists)
- Problem: Shortened feed content
 - Goose content extractor
- Evaluation: Case study

5. Conclusion


- Conclusion
 - Zeeguu Reader makes it possible to learn a new language in a comfortable way on Android devices
 - Includes planned features, still room for extensions
- Personal Lessons Learned
 - ORM: Comfortable way to implement database
 - Prioritize planned features
 - Gained experience in new programming languages
 - Performance optimization

The End

Questions?

Additional Material: ORM

```
22 @DatabaseTable(tableName = "feeds")    Database table
23 public class Feed {
24
25     // Id is generated by the database and set on the object
26     @DatabaseField(generatedId = true)
27     private int id;
28     ...
29     @DatabaseField(columnName = "favicon", dataType= DataType.BYTE_ARRAY)
30     private byte[] favicon;
31     ...
32     /*
33      * If eager is set to false then the collection is considered to be "lazy" and will iterate
34      * over the database using the Dao.iterator() only when a method is called on the collection.
35      */
36     @ForeignCollectionField(eager = false, orderColumnName = "date", orderAscending = false)
37     private ForeignCollection<FeedEntry> entries;    one-to-many
38
39     /*
40      * Only for read access, categories stored in this list do not get saved in the database!
41      * (Workaround because ormlite does not directly support m:m relations)
42      */
43     private ArrayList<Category> categories = new ArrayList<>();    many-to-many
```



The diagram consists of a large right-facing curly bracket on the right side of the code. The top part of the bracket groups lines 26 and 29, which are annotated with the text 'Database fields'. The bottom part of the bracket groups lines 37 and 43, which are annotated with the text 'one-to-many' and 'many-to-many' respectively.

Additional Material: ORM

- DAO Example

```
371     public void saveFeedEntry(FeedEntry entry) {  
372         try {  
373             if (entry.getId() == 0)  
374                 feedEntryDao.create(entry);  
375             else {  
376                 feedEntryDao.update(entry);
```

- Query Example

```
132     public List<FeedEntry> getRecommendedEntries(float maxDifficulty) {  
133         try {  
134             return callback.getDatabaseHelper().getFeedEntryDao().queryBuilder()  
135                 .orderBy("zeeguu_difficulty_average", true)  
136                 .where().between("zeeguu_difficulty_average", 0, maxDifficulty)  
137                 .query();  
138         }  
139         catch (SQLException e) {
```

Additional Material: ORM

- Schema upgrade

```
79  @Override
80  public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource, int oldVersion, int newVersion) {
81      try {
82          Log.d(DatabaseHelper.class.getName(), "onUpgrade");
83
84          // Drop the old tables
85          TableUtils.dropTable(connectionSource, Category.class, true);
86          TableUtils.dropTable(connectionSource, CategoryFeed.class, true);
87          TableUtils.dropTable(connectionSource, Feed.class, true);
88          TableUtils.dropTable(connectionSource, FeedEntry.class, true);
89
90          // After we drop the old databases, we create the new ones
91          onCreate(db, connectionSource);
92
93      }
94      catch (SQLException e) {
```

Additional Material: ORM

```
9  /**
10   * Database class to allow a many-to-many relation between categories and feeds in ormlite
11   */
12  @DatabaseTable(tableName = "category_feed")
13  public class CategoryFeed {
14
15      /**
16       * This id is generated by the database and set on the object when it is passed to the create method. An id is
17       * needed in case we need to update or delete this object in the future (ormlite does not support multiple
18       * primary keys).
19       */
20      @DatabaseField(generatedId = true)
21      private int id;
22
23      // This is a foreign object which just stores the id from the Category object in this table.
24      @DatabaseField(foreign = true, columnName = "category_id", columnDefinition = "integer references categories(id) on delete cascade")
25      Category category;
26
27      // This is a foreign object which just stores the id from the Feed object in this table.
28      @DatabaseField(foreign = true, columnName = "feed_id", columnDefinition = "integer references feeds(id) on delete cascade")
29      Feed feed;
30
31      CategoryFeed() {
32          // Empty constructor needed by ormlite
33      }
34
35      public CategoryFeed(Category category, Feed feed) {
36          this.category = category;
37          this.feed = feed;
38      }
39  }
```


Additional Material: WebView

- Zeeguu WebView
 - Extended Android WebView
 - Allows translation & bookmarking of words
 - Injects JavaScript in every webpage
 - JavaScript to Java Interface
- How does it work?
 - Word selection extension
 - Submit word for translation
 - Bookmark: Extract context
 - Highlight bookmarked word(s) using regex

Additional Material: Article R.

```
621 difficulties = []
622 for text in texts:
623     # Calculate difficulty for each word
624     words = util.split_words_from_text(text['content'])
625     words_difficulty = []
626     for word in words:
627         ranked_word = RankedWord.find_cache(word, language)
628
629         word_difficulty = 1.0 # Value between 0 (easy) and 1 (hard)
630         if ranked_word is not None:
631             # Check if the user knows the word
632             try:
633                 known_propability = known_probabilities[word] # Value between 0 (unknown) and 1 (known)
634             except KeyError:
635                 known_propability = None
636
637             if personalized and known_propability is not None:
638                 word_difficulty -= float(known_propability)
639             elif ranked_word.rank <= rank_boundary:
640                 word_frequency = (rank_boundary - (
641                     ranked_word.rank - 1)) / rank_boundary # Value between 0 (rare) and 1 (frequent)
642                 word_difficulty -= word_frequency
643
644         words_difficulty.append(word_difficulty)
```

Additional Material: Evaluation

- Case study
 - Mircea as participant
 - 9 articles from different difficulty levels
 - Video recording, “think aloud”
- Analysis
 - Understanding
 - Time per character
 - Percentage of words looked up
 - Percentage of words bookmarked

Additional Material: Evaluation

- Results (Average for difficulty groups)

Score	Understanding	Time per char	Looked up	Bookmarked
Easy (0.24)	4.50	0.21 s	6.52 %	5.19 %
Medium (0.32)	3.33	0.23 s	7.75 %	6.81 %
Hard (0.44)	2.66	0.28 s	11.13 %	7.92 %

Additional Material: Case Study

	O	P	Q	R
	Understanding	Time per Char	Percentage of words looked up	Percentage of words bookmarked
	4.50	0.237906423	7.774390244	6.25
	4.50	0.170936296	4.516129032	3.870967742
	4.50	0.227593152	7.272727273	5.454545455
⚡	1.50	0.232	8.860759494	7.911392405
	4.00	0.185257032	4.733727811	4.142011834
	4.50	0.25974026	9.653916211	8.378870674
	1.00	0.318133616	19.38534279	13.23877069
⚡	4.00	0.183260611	5.778894472	5.527638191
	3.00	0.332525742	8.214285714	5
	4.5	0.21214529	6.521082183	5.191837732
	3.333333333	0.225665764	7.749467839	6.810758304
	2.666666667	0.277973323	11.12617433	7.922136292