



Michael Öhninger, BSc

Implementation of a management information system based on analysis and presentation of user data

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Slany

Institute of Software Technology

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Slany

Graz, April 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Acknowledgments

A special thank goes to Jochen Kerschenbauer and Martin Maritsch who were supporting me during my whole master thesis.

I would also like to thank my family who is standing behind me all the time and my friends.

Abstract

This thesis describes the implementation of a management information system based on analysis and presentation of user data. The work within this project pursues the goal to provide a platform for football clubs to keep in touch with their fans easily. The responsible persons of clubs can provide a live ticker and create questions about a specific football game via a web-moderation application and fans can answer these questions via an Android- and iOS-App. Before this work has been done there was no possibility to find out specific details of a football game of the last weekend, e.g. how active a responsible person of a club has moderated a game in the frontend. In order to fulfil the challenges of a growing start-up and improve the overall experience of the product, a lot of activities must be logged, and periodic automatic analyses must be done.

In this master thesis the methodology of “Iterative and Incremental development” was applied. The aim of this methodology is to present one component after another component and not showing all in one to the customer. The whole system is finished when the customer agrees to all components. To achieve this, time series and moderator activities are logged. Developed automatic analysis of this thesis include an emergency list (live analysis of all football games), Facebook statistics (mentions of this project on Facebook) and a news list (news articles per club per week). This information is published for all clubs via a web-moderation application and project intern via Google Drive.

The responsible persons of football clubs can see interesting statistics like time series of their followers and a count of the clicks on hyperlinks of their sponsors. Additionally, project intern, time series of the followers of all teams, the above-mentioned analyses and a lot of other reports are presented.

With this thesis, club employees can draw conclusions. It is clear that they have done something wrong, when the followers or the users in the ranking during a football game are falling. Another possible outcome of the delivered statistics could be contracts with sponsors, when a specific club has very high access figures. For the project, a lot of processes are executed automatically. Consequently, person hours can be economised.

Kurzfassung

Diese Arbeit beschreibt die Implementation eines Management - Informationssystems durch die Analyse und Darstellung von Userdaten. Diese fand im Rahmen eines Projektes statt, welches das Ziel verfolgt, eine Plattform zu bieten, bei der Sportvereine sehr leicht mit ihren Fans in Kontakt treten können. Dabei ist es den Verantwortlichen von Fußballvereinen möglich, ein Spiel live zu tickern und selbst erstellte Fragen zu einem Spiel über eine Web-Moderatorenansicht zu stellen und die Fans können diese in einer Android- und iOS-App beantworten. Bisher gab es allerdings keine Möglichkeit, spezifische Details von Fußballspielen des vergangenen Wochenendes herauszufinden, z.B. wie aktiv ein Vereinsmoderator ein Spiel getickert hat. Um also den Herausforderungen eines wachsenden Startups gerecht zu werden und das Gesamterlebnis des Produktes zu stärken, sind regelmäßige Auswertungen erforderlich, die automatisiert geschehen.

In dieser Masterarbeit wird ein inkrementelles Vorgehensmodell angewendet. Das Ziel dieser Methodologie ist es, dem Kunden eine Komponente nach der anderen zu präsentieren und nicht alle auf einmal. Das ganze System ist dann abgeschlossen, wenn der Kunde allen Komponenten zustimmt. Um dies zu erreichen, werden Zeitreihen und Aktivitäten von Moderatoren mitgespeichert. Bei den automatisierten Auswertungen handelt es sich unter anderem um eine Notfalldienstliste (Live-Auswertung aller Fußballspiele), Facebook-Statistiken (Projekterwähnungen auf Facebook) und eine Newsliste (eingetragene Newsartikel der Vereine pro Kalenderwoche). Diese Information wird entweder allen Clubs über eine Web-Moderatorenansicht oder projektintern über Google Drive zur Verfügung gestellt.

Die Verantwortlichen von Fußballvereinen können interessante Statistiken wie Zeitreihen der Followers ihres Teams und die Anzahl der Klicks auf Sponsorenlinks sehen. Zusätzlich werden projektintern Zeitreihen aller

Teams, die oben erwähnten Auswertungen und eine Menge anderer Statistiken präsentiert.

Mithilfe dieser Arbeit können Mitarbeiter eines Vereins Schlüsse über ihre Arbeit ziehen. Es ist klar, wenn die Followers oder die Benutzer in der Spielrangliste sinken, dass irgendetwas falsch gemacht wurde. Es könnten auch Verträge mit Sponsoren entstehen, wenn ein Fußballklub hohe Besucherzahlen aufweist. Innerhalb des Projektes werden einige Prozesse automatisiert. In weiterer Folge können Personenstunden eingespart werden.

Contents

Abstract	iv
Kurzfassung	vi
1. Introduction and overview	1
1.1. Context of this thesis	1
1.2. Situation analysis	2
1.3. Practical and scientific relevance	5
1.4. Methodology	6
1.5. Aim of this thesis	7
1.6. Challenges of this thesis	8
1.7. Structure of this thesis	11
2. State of the Art	12
2.1. Methodology	12
2.1.1. Waterfall Development	12
2.1.2. Spiral development	14
2.1.3. Iterative and Incremental development	15
2.2. Cloud services platform	17
2.3. Real time stream processing	20
2.4. RESTful API	23
2.5. NoSQL Databases	24
2.6. Frontend Framework	26
3. Implementation	29
3.1. Infrastructure	29
3.1.1. Analytics topology	29
3.1.2. Clickstream user data	33

Contents

3.2. Log data	35
3.2.1. RESTful API	36
3.2.2. Clickstream user data	39
3.2.3. Time series	41
3.2.4. Month rankings	45
3.2.5. Game specific details	45
3.3. Analyse data	49
3.3.1. RESTful API	49
3.3.2. Serverless Interactive Query Service	52
3.3.3. Social media analysis	53
3.3.4. Moderator statistics	57
3.3.5. Game statistics	58
3.4. Display data	58
3.4.1. Internal communication	59
3.4.2. External communication	65
4. Results	69
4.1. Reports	69
4.2. Frontend	86
5. Discussion	89
5.1. Reports	89
5.2. Frontend	93
6. Conclusions	96
6.1. Evaluation of the Research Questions	96
6.2. Limitations	98
6.3. Future work	98
A. Logging in the app	102
Bibliography	110

List of Figures

2.1. Waterfall model	13
2.2. Iterative and Incremental development	16
2.3. Example of a topology which contains spouts and bolts	22
3.1. Example of topologies	30
3.2. Spouts of the analytics topology	31
3.3. Bolts of the analytics topology	32
3.4. Data flow to log clickstreams	32
3.5. Data flow to log football games	32
3.6. Data flow to log football game rankings	33
3.7. Data flow to log followers	33
3.8. Data flow to log rankings	33
3.9. Analytics topology in Apache Storm	34
3.10. Data Flow in Amazon Athena	35
3.11. GET request for receiving followers within a time range	51
3.12. Example query in the GUI of Amazon Athena	54
3.13. Gain Facebook timeline of a team via Graph API	57
3.14. Gain Facebook fan count of a team via Graph API	58
3.15. Generated slack messages	66
4.1. Google Drive folders	71
4.2. News report (Google Spreadsheet)	72
4.3. Future games report (Google Spreadsheet)	73
4.4. Matchday analysis (Google Spreadsheet)	75
4.5. Facebook report (Google Spreadsheet)	78
4.6. Rankings report (Google Spreadsheet)	80
4.7. Emergency list report (Google Spreadsheet)	84
4.8. Angular 5 application displaying time series of followers	88

List of Tables

3.1. Output of an Amazon Athena query to receive the Android and iOS viewed news articles count	53
A.1. Logged app actions	102
A.2. Logged menu actions	103
A.3. Logged notification actions	103
A.4. Logged deep link actions	103
A.5. Logged settings actions	104
A.6. Logged news actions	104
A.7. Logged event actions	105
A.8. Logged question actions	105
A.9. Logged post actions	106
A.10. Logged activity actions	106
A.11. Logged team actions	107
A.12. Logged sponsor actions	107
A.13. Logged league actions	108
A.14. Logged ad actions	108
A.15. Logged info actions	109

List of Listings

3.1. Promises in Nodejs (Promise.all)	36
3.2. Promises in Nodejs (sequential execution)	37
3.3. Example of a log activity of a moderator (entering a line-up)	37
3.4. Example of a log entry	40
3.5. News views	40
3.6. Sponsor clicks	41
3.7. Log entry of event ranking	42
3.8. Log entry of league followers	44
3.9. Log entry of team followers	44
3.10. Amazon Athena query to get all Android, iOS and overall news view counts since a specific date	52
3.11. Gain Facebook timeline of a public accessible page	54
3.12. Classify shared hyperlinks on Facebook	56
3.13. Gain Facebook fan count of a public accessible page	56
3.14. Get specific file id in Google Drive	59
3.15. Move file to a specific folder in Google Drive	60
3.16. Generating emails	63
3.17. Generating slack messages	65
3.18. Angular highchart	67

1. Introduction and overview

In this chapter, a general overview about the topic of this thesis is given and the scientific relevance will be discussed.

This master thesis describes the work in a start-up which is on the way to become a company.

1.1. Context of this thesis

This project provides a platform to improve the communication between football fans and clubs and is the first club-to-fan platform in Austria, where club employees can ask and be in contact with their fans. Club moderators have the possibility to post news, ask questions and deliver a live ticker of a football game to fans. They could also reach their fans Android and iOS-based smartphones directly through notifications.

The whole software project is separated into four areas:

- Backend: Infrastructure based on Amazon Web Services, a JavaScript framework which runs on server-side (Node.js), a real-time computation system (Apache Storm) and a NoSQL database based on MongoDB.
- Frontend: Represents a responsive web application based on AngularJS, newly developed in Angular 5.
- Android-App: Developed for smartphones with the operating system of Google.

1. Introduction and overview

- iOS-App: Developed for smartphones with the operation system of Apple.

There are also people who are responsible for marketing, sales and support.

1.2. Situation analysis

In March 2017, this project went online and there is a cooperation with amateur clubs, especially located in Styria, since then. There are several thousand Android and iOS app users at the moment, but currently there is no way to analyse this data in a structured way.

1. Problem

At the moment there is no possibility or only one with large effort to research specific details of football games of the past weekend, e.g. how active was a football game moderated by a club or generally how many teams, which have a cooperation with this project, had a game on the last weekend. It is also impossible to check when and how app users handle the app.

Here are other questions listed that cannot be answered at the moment:

- How many people open the Android- and iOS app in one week, month and year?
- Which user has collected the most points in a specific league in the last month by answering questions before, during and after a football game?
- Did the followers count of a team or league decreases or increases during the last weeks and months?

There are also no statistics about football games of each round per league, state and country.

Research topic

After this thesis the following things should be logged and researched:

1. Introduction and overview

a) Teams

- Moderator
 - What did a moderator specifically during a football game? (in relation to game activities, questions, posts, match sponsor changed, questions evaluated, entered line-up)
 - Which moderators have written a lot of news articles, shared project-related things on Facebook and moderated a football game very well in the frontend? After this question is answered, the best club moderators can get honoured and afterwards they will be more motivated in future to do things on this platform.
- App users
 - Time series of users in ranking and observers of a football game
 - Time series of the followers count of a team
- Statistics
 - Statistics about clicks on news articles and sponsor clicks
 - Clickstream analysis: Contains all actions of Android- and iOS app users.
 - Find interesting coherences within a team, e.g. by comparing home and away goals or first half and second half goals.
 - Find out how active a team supports this project on Facebook: Contains mentions, shared links, shared attachments and all reactions of fans to their postings.

b) Leagues

- App users
 - Time series of users in ranking and observers of a league
 - Time series of the followers count of a league
 - Rankings: Deliver month and season rankings of the app users who have collected the most points by answering questions before, during and after a football game.
- Statistics
 - Deliver football statistics within each round, e.g. by comparing home and away goals.

1. Introduction and overview

- Find interesting coherences of a league, e.g. with formation 4-4-2 eight out of ten teams won their last football game.

2. Problem

In conclusion to the first problem there exists no management information system at the moment, where the delivered data can be published internally, e.g. the evaluation of all football games of the last weekend. A very useful tool to interpret an emergent trend are time series of the followers count of a team. When the followers count is falling, the team probably used this platform rarely lately. With such a management information system it will be possible to detect problems and make clear statements about the usage of this platform by a specific football team. Problem clubs can be figured out very fast.

After creating such an information system, it would be also easy to indicate a moderator of a football game, when he uses the frontend inaccurate or to provide improvement suggestions for using this platform optimal.

There is also a differentiation between internal and external communication. For all people of this project, this data must be published internally and for the communication with a football team, an external communication channel is needed.

- Internal communication: Google Drive and Slack
Deliver reports like the summary of football games of the last weekend, month rankings with the best app users, team and league rankings with the best app users, team statistics and a lot of other interesting reports.
Slack is used as a channel within this project team to communicate with each other.
- External communication: Angular application
There exists already a web application within this project. In the moderation view moderators of football clubs can handle football games, news articles, sponsors, squad and a lot of other things. In the administration view football teams, leagues, question and post templates within a football game and users can be handled. The above-mentioned data should be integrated in the administration and moderation view, e.g. time series are a very useful

1. Introduction and overview

tool to make historical and predictive analyses of follower counts and game rankings.

1.3. Practical and scientific relevance

The following scientific questions will be answered during this master thesis:

1. “How and with which components is it possible to save a great amount of user data and analyse it under the usage of the current State of the Art?”

With clickstream analysis of the Android- and iOS app, a lot of data can be collected. After logging this data, it must be analysed to draw conclusions. Another data that should be logged are activities of moderators of football clubs before, during and after a game.

Useful user data include the followers count of all teams in the database and logged football game rankings (with users in ranking, viewers, anonymous users in ranking and visits). Other rankings like team, league, state and country rankings can also be logged.

After this it can be established when and what a moderator has done in the frontend and how many app users have clicked on a sponsor or a news article in the Android and iOS app. With all this logged data there are many ways to find out specific details of the usage of this platform by moderators and app users.

2. “How is it possible to save time series and display them with time series analysis?”

The following data will be logged:

- Followers of teams and leagues: Contains the followers count of each team in the database.
- Game rankings: Contains users in ranking, viewers, anonymous users in ranking and visits of the Android and iOS app.
- Team, league, state and country ranking: Contains users in ranking and anonymous users in ranking.

1. Introduction and overview

This data must be logged in a database in an optimal way to display it later. This information should be displayed in a web application to observe trends. In further consequence it should be possible to analyse this logged data.

The practical relevance of this problems is to conclude from the data how users and club moderators use this platform. This information can be used to receive money from sponsors for displaying their names at different times of the day and at different times of a football game.

1.4. Methodology

In Section 2.1, three possible software development processes are mentioned and described. In this master thesis the methodology of “Iterative and Incremental development” was applied.

This method in software development is characterized by having a large number of components. When one component is finished the feature will be presented to the customer. After receiving feedback of the customer, the component can be improved. In further consequence this methodology reduces development time.

The aim of this methodology is to present one component after another component and not showing all in one to the customer. The whole system is finished when the customer agrees to all components.

As explained in Choetkiertikul et al. (2017), iterative software development is widely practiced in industry. It is essential to monitor the execution of an iteration. Modern software development often uses this methodology and this model has essential parts in many methodologies such as Unified Process, Extreme Programming, Scrum and other agile software development methods.

1.5. Aim of this thesis

The end situation of each stakeholder of this project after this thesis is described in this section.

There are three stakeholders:

- Teams
After the completion of this project it should be possible for a user of the web application to detect a trend after analysing the logged data. The club employees can then draw conclusions out of this information. It is clear that club moderators have done something wrong, when the followers count or the users in the ranking during a football game are falling. In further consequence, they will know their fans better. Another possible outcome of the delivered statistics could be contracts with sponsors, when a specific club has very high access figures. In that case advertisements of sponsors of a team can be displayed.
- Fans
The club staff can detect possible problems in the usage of this platform, when they recognise that the followers count of their team is falling. Consequently, the fans will be better entertained by the provided information about the club and during football games, when the club staff has drawn direct conclusions out of the delivered statistics.
- This project
At the moment it is not possible or only with huge time investment to detect the mentioned things about app users, moderators, teams, leagues, states and countries. Sometimes backend developers of this project have to look manually in the database to detect what a moderator has done before, during and after a football game. With this logged information it will be possible to react to criticism and understand improvement suggestions of football clubs or fans. It is also possible to reply faster in the case teams use this platform not as intended and deliver hints to make the overall experience better.

1. Introduction and overview

Another possible outcome could be contracts with sponsors, when there are high access figures at the moment and in further consequence, advertisements are displayed.

1.6. Challenges of this thesis

In this section the challenges of this thesis will be presented.

Ad aim 1: Gather team- (moderator, player, statistics) and league (player, statistics) specific data

- Requirement analysis
- Establish an infrastructure: Provide a route for the Android- and iOS app to log specific occurring events.
- Gather data out of different data sources:
 - For the analysis of a football game the already existing database contains all needed information.
 - Other information must be logged:
In many cases the recorded activities include things moderators do during a football game in the frontend, e.g. enter goals, enter the line-up, answer questions, start the match etc.
In order to log such activities, the name of the moderator of the team, a timestamp and the activity of the moderator will be saved, because this information is unknown yet.
- Analyse data
 - During a football game
 - Deliver football statistics about leagues, states and countries

It is important to start the analysis automatically not until a specific event (e.g. the end of the football game) occurs.

Ad aim 2: Provide this information to the teams or project internally either through Google Drive or a web application and display this information

- Requirement analysis

1. Introduction and overview

- Internal communication: Provide data through the Google Drive Account of this platform:
 - Creation of Google Spreadsheets through Sheets API.
 - Uploading this file through Drive API to a specific folder.
- External communication:
 - Investigate State of the Art frameworks for time series analysis.
 - Integration of the logged data in an existing Angular application.

Integration in the existing web application

There exists already a so-called administration view, where the above-mentioned statistics should be displayed on following places:

- Project internally:
 - Statistics menu item should appear on the right to league management.
 - Time series of global ranking per year (users in ranking / viewers / anonymous users).
 - Provide hyperlinks to specific Google Drive folder:
 - * Leagues- and team rankings report: Contains all fans which have collected the most points in the current season.
 - * Month rankings report: Contains all fans which have collected the most points in the last month.
 - * News report: Contains all written news articles per calendar week with access figures.
 - * Future game report: Contains all entered future games until the next week.
 - * Game statistics report: Contains interesting coherences of game statistics within a league.
 - * Matchday analysis: Contains information about matches and is delivered one day after a match takes place.
 - * Emergency report: Contains live information about games of today and the day before.
 - * Facebook statistics report: Contains all shared Facebook postings in relation with this project and all reactions of fans.

1. Introduction and overview

- * Team statistics report: Contains general team information and rates the activities of team moderators in relation to Facebook posts, news articles and activities on matchdays.
- * Sales report: Contains contact details to each team which is part of this project.
- Menu item team management:
A new symbol should appear on the left to the edited symbol of each team:
 - * Time series of team followers
 - * Time series of team rankings (users in ranking / viewers / anonymous users)
- Menu item league management:
A new symbol should appear on the left to the edited symbol of each league:
 - * Time series of league followers
 - * Time series of league rankings (users in ranking / viewers / anonymous users)
- Visible for all club moderators of all teams:
 - Statistics menu item below the squad menu item should appear with following tabs:
 - * General statistics:
 - Time series of team followers
 - Time series of team ranking (users in ranking / viewers / anonymous users)
 - * Sponsor: Contains the information of click counts on a hyper-link of a sponsor.
 - * News: Contains the information about click counts on a news article.
 - Menu item games:
A new symbol should appear on the left to the sharing symbol:
Time series of game ranking (users in ranking / viewers / anonymous users / visits)

1.7. Structure of this thesis

In Chapter 2 an overview about the current State of the Art of methodologies, the infrastructure and the used software is given, where the possibilities in infrastructure and software tools are discussed.

Chapter 3 describes the used infrastructure and the implementation of logging, analysing and displaying user data.

The results of this master thesis will be presented in Chapter 4.

Afterwards, in Chapter 5 the outcome of the work will be discussed.

In the last Chapter 6, conclusions will be mentioned, and future work will be discussed.

2. State of the Art

In order to follow the principles of current methodologies, this topic is examined in this Chapter.

This project's infrastructure, backend and frontend use a set of modern technologies to supply its functionality. In this chapter the State of the Art of this project will be presented.

2.1. Methodology

In this section current methodologies in the software development process and their life cycle will be described.

2.1.1. Waterfall Development

The waterfall model is one of the oldest software development models. It has many weaknesses but is currently in use in many software projects.

The waterfall process is executed sequentially. The output of each phase is the input of the next phase. The next phase cannot start until the previous phase is completed. There is the possibility of a small overlapping of a phase with their next phase. Typically, there are entrance and exit criteria in each phase in order to control everything. The whole process can be seen in Figure 2.1.

After the requirements phase, unforeseen issues with the design can arise. For such problems some requirements can be deleted or changed in order to fulfil all requirements.

2. State of the Art

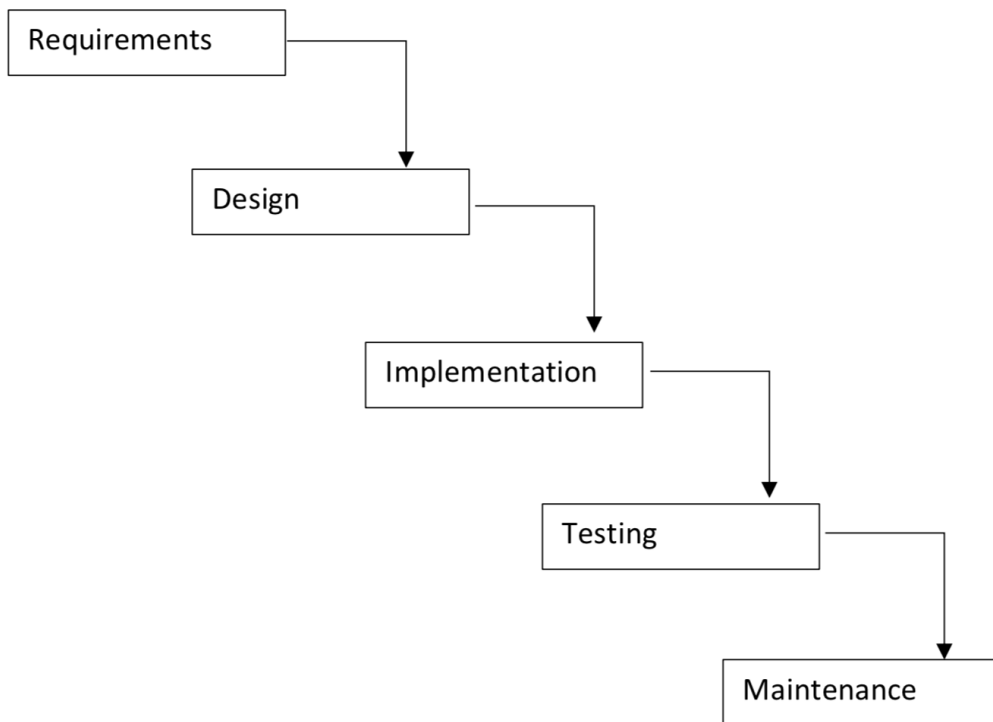


Figure 2.1.: Waterfall model

2. State of the Art

Another problem exists between testing and maintenance, e.g. when customers detect errors in released software. In this case new test cases must be written and the also the phase maintenance must be updated to cover all new test cases.

A major issue of the waterfall model is the occurrence of the testing phase almost as the last phase of this model. When performance or storage problems in the software development process are detected very late, probably it is required to update also all other phases of this model.

Advantages:

- Simple and easy to use: All phases are processed sequentially with the possibility of a small overlapping between each phase.
- Exists since many years: This model is generally accepted.
- Works well for smaller projects: When all requirements are known at the beginning it works very well.

Disadvantages:

- Requirements must be known at the beginning: Before the whole process starts, all features, that should be developed, must be known.
- Major problems are discovered very late.
- Lack of parallelism: Each phase is implemented after its previous phase.

The source of this subsection about the waterfall development is Braude and Bernstein (2016).

The paper of Chari and Agrawal (2017) has correlating conclusions. After a sample of 49 software projects, which follow the waterfall methodology, was analysed it was found that change requests based on incorrect requirements increases the number of new requirements as well as the number of defects in the software.

2.1.2. Spiral development

Spiral Model is developed by Barry Boehm, it is called spiral because Boehm developed it as an outward spiral.

2. State of the Art

This software development model is a risk-driven model. Each cycle has the aim to increase the degree of system definition and decrease the degree of risk. Risk management is a very huge topic in the whole process.

Each iteration contains the following steps:

1. Identification of critical goals of the project.
2. Evaluation of alternatives of the process to reach goals.

Advantages:

- Risks are managed during the whole process.
- Errors are eliminated early.
- Planning is part in each phase.

Disadvantages:

- Complicated to use: Expertise is required in this software development process. model
- Too much effort for small projects: It is needless, when risk analysis takes a huge part in the process.

The source of the information in this subsection is Braude and Bernstein (2016).

2.1.3. Iterative and Incremental development

As described in Braude and Bernstein (2016), a major disadvantage of the waterfall model is the linear fashion and except for smaller projects this is impractical. It is needed to revisit all phases in the development process.

In the iterative and incremental development, all phases are revisited in a cyclical manner. A part of the system is developed and tested, after receiving feedback more of the system is implemented. With such a system it is not needed to understand everything at the beginning and not all features must be known before the implementation starts.

An iterative process is defined as repeated execution of the waterfall model and it is an incremental process, when the tasks in an iteration are relatively small. In Figure 2.2 such a process with three iterations can be seen.

2. State of the Art

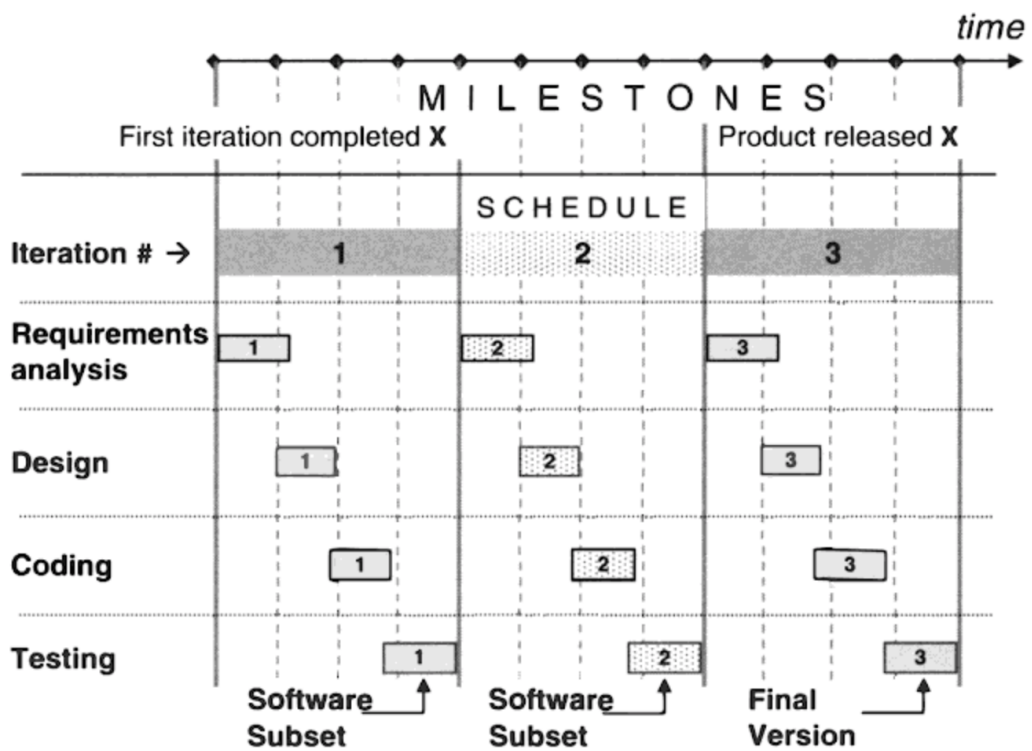


Figure 2.2.: Iterative and Incremental development (Source: Cockburn, Alistair. "Unraveling incremental Development" January 1993, <http://alistair.cockburn.us/Unraveling+incremental+development>)

2. State of the Art

Regression testing is very important in this methodology and ensures that previously developed software still acts the same way as it acts before a change was made. As described in Rosero, Gomez, and Rodriguez (2017), regression testing helps to ensure the reliability in the process of building the software and helps to verify existing modifications (fixing bugs) or verify new features added to a software product.

2.2. Cloud services platform

With Cloud computing it is possible to have access to servers, databases, storage and a lot of other services. This project uses Amazon Web Services as cloud services platform.

Six advantages of cloud computing are listed in Amazon Web Services (2018[b]):

- Variable costs instead of fixed costs: With such a platform you only have the variable costs instead of investing a lot of money in data warehouses at a time, where it is not clear, what is needed.
- Huge economies of scale: Amazon Web Services are highly used by many large companies. With the use of this services there are lower costs than running services alone.
- Not estimating capacity sizes: It is very complex and costly to analyse the situation in order to get the capacities that are really needed. With Amazon Web Services it is easy possible to scale down, when there is not the need of such a number of servers, or to scale up, when a lot more capacities are needed.
- React fast to unexpected situations: It does not take weeks to speed up the performance of such services, it is possible within minutes to react to unexpected situations and be very flexible.
- Investing money in preserving data warehouses: With the outsourcing of such services it is not needed to maintain data warehouses and it is possible to focus on core competences of the company.
- Be fast available around the world: With only a few clicks it is possible to deploy applications around the world to fulfil local customer needs.

2. State of the Art

There exist a lot of products made available by Amazon Web Services. They can be categorised in the following areas and the used services in context of this thesis are accurately described. The source of this information about AWS products can be found in Amazon Web Services (2018[a]).

- Compute: Contains services like virtual servers, code can be executed after the occurrence of an event (AWS Lambda), batch jobs can be implemented, and a lot of other services are available.
- Storage: Includes services like block storage for virtual servers in the cloud, managed file storage for ECS and petabyte-scale data transport.
 - Amazon S3: Represents a scalable storage. There are a lot of advantages like durability, availability and scalability. It is also possible to perform queries in place. So, when saving data to Amazon S3, analytics tools like Amazon Athena (will be described later) can be executed directly on Amazon S3 and no interface is needed between them.
- Database: There are services like managed relational databases (Amazon Aurora), in-memory data store and cache (Amazon ElastiCache) and graph database service (Amazon Neptune).
 - Amazon Redshift: Represents a cost-effective data warehousing. It is possible to perform SQL queries and use Business Intelligence (BI) tools. Redshift Spectrum is also included in this service, where exabytes of unstructured data can be searched.
- Migration: AWS also offers migration tools like AWS Database Migration service (migrate databases with minimal downtime) and AWS Snowmobile (exabyte data transport).
- Networking & Content Delivery: There are also products like Amazon API Gateway (to build, deploy and manage APIs), Elastic Load Balancing (distributes incoming traffic across multiple targets) and AWS Direct Connect (network connection to AWS).
- Developer Tools: Includes tools like AWS Command Line Interface (to manage AWS products via command line interface), AWS CodeStar (develop and deploy AWS applications) and AWS CodeBuild (to build and test code).
- Management Tools: Contains products like AWS Trusted Advisor (to optimize performance and security) and Amazon CloudWatch

2. State of the Art

(monitor resources and applications).

- Media Services: There are services like Amazon Kinesis Video Streams (to process and analyse video streams) and AWS Elemental MediaLive (to convert live video content).
- Security, Identity & Compliance: Security is a highly interesting topic currently. To provide adequate security Amazon offers services like AWS Single Sign-On (to manage SSO access and user permissions), Amazon Inspector (to analyse application security) and AWS WAF (filter malicious web traffic).
- Analytics: Analytics tools are important and will be more important in future. AWS provides a lot of products to fulfil these requirements. A very useful tool is Amazon Quicksight, where visualizations can be built, and statistics can be found very fast. Other services are Amazon Athena, Amazon Kinesis and Amazon Redshift.
 - Amazon Athena: With Amazon Athena it is possible to use analytics mechanism to query S3 storage with SQL directly. It is possible to perform queries. Athena is based on Facebook Presto¹ and also includes Apache Hive².
 - Amazon Kinesis: It is classified as an analytics service. With Kinesis it is possible to work with real-time streaming data. More precisely data can be collected, processed and analysed. It can be used to log clickstreams of applications.
 - Amazon Redshift: The previously mentioned service Amazon Redshift (in category database) can also be used as an analytics tool.
- Machine Learning: There are useful and deeply interesting products like Amazon Translate (fluent language translation), Amazon Lex (build voice and text chatbots), Amazon Comprehend (discover insights and relationships in text) and Amazon Transcribe (automatic speech recognition).
- Mobile Services: Amazon also offers mobile services like AWS Mobile Hub (to build, test and monitor apps), Amazon Pinpoint (push notifications for mobile apps) and AWS Mobile SDK (a mobile software development kit).

¹<https://prestodb.io>

²<https://hive.apache.org>

2. State of the Art

- AR & VR: With Amazon Sumerian it is possible to build and run AR and VR applications.
- Application Integration: Consists of products like AWS Step Functions (to coordinate distributed applications), a queue service (SQS) and Amazon Simple Notification service (SNS).
 - Amazon Simple Queue Service (SQS): With this message queuing service no messages will be lost and can be handled individually.
- Customer Engagement: Contains cloud-based contact centre (Amazon Connect), push notifications for mobile apps (Amazon Pinpoint) and an email service (SES).
 - Amazon Simple Email Service (SES): With this email service it is possible to send and receive emails.
- Business Productivity: There are also products to increase the productivity like Alexa for Business (can be used to decrease tedious tasks at work) and Amazon WorkMail (secure and managed business email and calendaring).
- Desktop & App Streaming: There are services like Amazon WorkSpaces (providing a secure Desktop-as-a-Service (DaaS) solution) and Amazon AppStream 2.0 (streaming desktop applications to a browser included).
- Internet of Things: It is possible to use tools like AWS IoT Core (connect devices to the cloud), AWS IoT Analytics (provide analytics for Internet of Things devices) and AWS IoT Device Management (organize and remotely manage Internet of Things devices). There are also a lot of other services.
- Game Development: This category provides services like Amazon GameLift (game server hosting) and Amazon Lumberyard (cross-platform game engine).

2.3. Real time stream processing

As described in Iqbal and Soomro (2015), big data is a synonym for competitive advantages in business rivalries and one of the most trusted real time

2. State of the Art

processing and fault tolerant tool is called Apache Storm. Apache Storm is highly scalable and offers low latency. Apache Storm was presented on 17 September 2014 and provides a simple architecture to build applications called topologies.

With stream processing it is possible to process data. Data can arrive anytime and must be evaluated in real-time.

The paper of Veen et al. (2015) characterizes stream processing platforms as applications to analyse incoming data continuously. It is not easy to predict how much computing resources are needed for a specific use case, because the velocity of input data may change over time. In Apache Storm, all machines within an established cluster are used to fulfil this requirement.

Apache Storm (2018[a]) is a real-time computation system. It allows reliable processing of streams of data and the use of any programming language.

The concept Apache Storm (2018[b]) is separated in the following components:

- Topologies: The whole logic of stream processing is stored in topologies. Unless a topology is killed it runs forever. A topology is a graph consisting of spouts and bolts connected with stream groupings.
 - TopologyBuilder: This class is used to construct topologies in Java.
 - Running topologies on a production cluster: With StormSubmitter the created topology can be submitted to a cluster. The number of workers and the maximum spouts, which are pending, can be defined.
 - Local mode: It is also possible to test topologies locally.
- Streams: Nodes are connected through streams.
 - Tuple: Can be described as main data structure. It consists of a list of values with any type (integers, longs, shorts, bytes, strings, doubles, floats, booleans and byte arrays).
 - OutputFieldsDeclarer: Streams and their schemas can be declared.
- Spouts: Every spout can emit one or more emission streams. Streams must be declared with the “declareStream”-method of “OutputFieldsDeclarer” and the stream, where it has to emitted, has to be defined

2. State of the Art

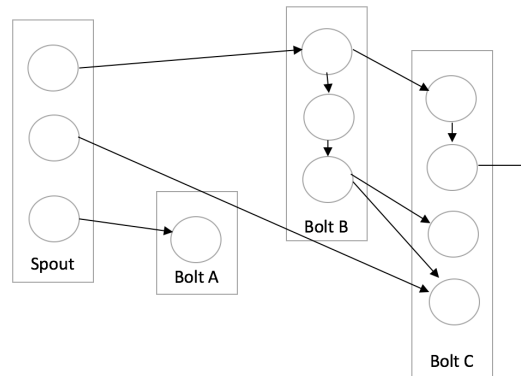


Figure 2.3.: Example of a topology which contains spouts and bolts

in the emit-method on “OutputCollector”. The main method is “next-Tuple”, where a new tuple is emitted into the topology or simply returns if there are no tuples to emit.

- Bolts: The challenges of bolts are to process tasks. Bolt can emit one or more emission streams like spouts.

An example of a topology can be seen in Figure 2.3. Spouts and bolts are represented as rectangles. One circle represents one worker, arrows indicate streams between workers. A bolt is able to push to a database.

In bolts, streams they should receive as input must be defined. The following stream groupings define how the stream should be partitioned:

1. Shuffle grouping: In this grouping, tuples are randomly distributed, but each bolt gets the same number of tuples.
2. Fields grouping: In this group, fields can be specified in the grouping and these fields will decide the grouping of the stream. When a stream is grouped by an id all tuples with the same id go to the same task.
3. Partial Key grouping: It is likely the fields grouping unless the fields are load balanced between two downstream bolts.
4. All grouping: Here the stream contains all tasks of each bolt.
5. Global grouping: Every tuple is processed by the same bolt instance.

There are different system components in Apache Storm:

2. State of the Art

- Nimbus: The nimbus represents the master node. Nimbus analyses the topology and distributes the founded task to a free supervisor. When a thread fails, the nimbus will manage to restart the thread on another node.
- Zookeeper: Apache ZooKeeper is a service for coordinating group of nodes and synchronize them. With ZooKeeper it is possible to interact with the nimbus and it can detect the state of nimbus and supervisor.
- Supervisor: The supervisor describes the worker node and has multiple worker processes. They follow instructions of the nimbus.

Apache Storm is written in Java and its source code is open-source. Spouts and bolts can be defined in any programming language like Ruby, Python and JavaScript.

2.4. RESTful API

REST³ uses HTTP⁴ as a protocol. In the context of this project, the RESTful API⁵ is based on Express⁶, a web framework for Node.js⁷.

As stated in Tilkov and Vinoski (2010), Node.js - also called Node - is a server-side JavaScript environment and is based on Google's runtime implementation named "V8" engine. Node does not rely on multithreading to support concurrent execution of business logic.

Node.js (2018) is an asynchronous event driven JavaScript runtime. With Node.js it is possible to build applications with network access.

As described in Tilkov and Vinoski (2010), JavaScript is the perfect language for this approach, because it supports callbacks. After a specific event occurs, it triggers a callback.

There are four HTTP methods that are used in such an architecture:

³REST ... REpresentational State Transfer

⁴HTTP ... Hypertext Transfer Protocol

⁵API ... Application Programming Interface

⁶<http://expressjs.com>, visited on 03/13/2018

⁷<https://nodejs.org/en/>, visited on 03/13/2018

2. State of the Art

- GET: Provides read only access to a resource.
- PUT: Provides the possibility to create a new resource.
- DELETE: A resource can be deleted.
- POST: With the POST-method a new resource can be created or an existing resource can be updated.

The package manager of JavaScript and Node.js is npm⁸. With npm it is possible to discover packages of reusable code and it provides powerful functionalities.

Chanotis, Kyriakou, and Tselikas (2014) shows that Node.js offers client-server development integration and is the perfect tool for developing fast, scalable network applications. It outperforms Nginx and Apache in input/output (I/O) operations.

Another highly interesting tool for API development is Postman⁹. With Postman it is possible to perform HTTP requests and test the self-developed routes.

2.5. NoSQL Databases

As described in Leavitt (2010), many organizations collect huge amounts of customer, scientific and a lot of other data for future analysis. Traditionally, this data was stored in relational databases.

Developers have begun using non-relational databases, now called NoSQL¹⁰. Such databases can handle unstructured data. Proponents of NoSQL databases enable better performance, which is very important for applications with huge amounts of data.

The source of the following information is MongoDB (2018[b]):

NoSQL uses a set of different database technologies in further consequence of modern applications:

⁸<https://www.npmjs.com>, visited on 03/13/2018

⁹<https://www.getpostman.com>, visited on 03/13/2018

¹⁰NoSQL ... Not only Structured Query Language (SQL)

2. State of the Art

- It is needed to change data types very often and handle a huge amount of data not only as structured, but also as semi-structured, unstructured and polymorphic data.
- Today software developers must perform in agile sprints, iterate quickly and push code every week. Once software was developed in a waterfall development cycle.
- Applications are now served as services that are always online, accessible around the world, and not only once served to the customers.
- Organizations today tend to use cloud computing, open source software and not maintaining large server infrastructure.

Relational databases are not able to match up with these new requirements.

Here an overview about NoSQL Database types is given and can be found in MongoDB (2018[b]):

- Document databases: There exists a key in combination with a data structure called document. Documents can contain nested documents, a variety of data types and arrays. MongoDB is an example for such a database.
- Graph stores: With graph stores information about networks of data like social connections can be stored. Examples of this database type are Neo4J and Giraph.
- Key-value stores: The simplest version of NoSQL databases. Every item in the database is stored as a key, together with the value.
- Wide-column stores: Examples are Cassandra and HBase. They are optimized for queries over large datasets.

As described, MongoDB¹¹ is a NoSQL database. Queries in MongoDB have the following components:

- Collection: The collection which should be queried.
- Comparison query operators: The following operators for comparing data are available:
 - \$eq: Checks equality of two values.
 - \$gt: Checks if the first value is greater than the second value.

¹¹<https://www.mongodb.com>, visited on 03/13/2018

2. State of the Art

- \$gte: Checks if the first value is greater or equal than the second value.
- \$in: Checks if a value is contained in an array.
- \$lt: Checks if the first value is less than the second value.
- \$lte: Checks if the first value is less or equal than the second value.
- \$ne: Checks if the first value is not equal to the second value.
- \$nin: Checks if a value is not contained in an array.
- Logical query operators:
 - \$and: Selects all documents where all defined expressions evaluate to true.
 - \$not: Selects all documents where the defined expression does not match.
 - \$nor: Selects all documents where all defined expressions fail.
 - \$or: Selects all documents where one of the defined expressions must be evaluated to true.
- SelectFields: When only specific attributes of the queried collection are needed and not all attributes.
- SortByFields: When the result must be sorted ascendant or descendent.

A very useful tool to test queries is NoSQLBooster¹² for MongoDB (formerly MongoBooster). It is a cross-platform GUI¹³ tool for MongoDB to query data and perform the four basic operations create, read, update, and delete (CRUD) in NoSQL databases.

2.6. Frontend Framework

With frontend frameworks, web applications can be build. As can be seen in C Sharp Corner (2018), web development includes client and server side. Client-side frameworks are typically written in JavaScript and run in web browser. Examples are Angular, React and Vue. According to the article of

¹²<https://nosqlbooster.com/home>, visited on 03/13/2018

¹³GUI ... Graphical user interface

2. State of the Art

C Sharp Corner (2018), the Top 5 trending front-end frameworks in 2018 are Angular, React, Backbone, Ember and Vue.

In this thesis, Angular¹⁴, the successor of AngularJS 1 and AngularJS 2, was used as frontend framework. Angular (2018[a]) is a framework for building client-side web applications based in HTML. It is written in either JavaScript or TypeScript (can be compiled to JavaScript). Angular is one of the most popular web development frameworks.

As stated in Evans (2017), it aims to simplify both the development and the testing of such applications. It is based on a client-side MVC¹⁵ and MVVM¹⁶ architecture.

This framework is based on components that can be reused very easily. The logic is added in so-called services. The components and services must be added in modules.

The most important Angular tools are stated in Angular (2018[b]):

- Angular CLI: This tool makes it easy to create an already functioning application, to generate components and test the app locally with a simple command.
- Angular Playground: Provides an open source for building, testing and documenting Angular applications.
- Angular Universal: Server-side rendering for Angular applications.
- Celerio Angular Quickstart: It is a tool for generating Angular applications from an existing database scheme.
- Codelyzer: Provides an analysis for Angular applications.
- Compodoc: This tool provides the possibility to generate documentations for Angular applications.
- Lite-server: This tool serves a web application and automatically updates, when HTML¹⁷ or JavaScript code has been changed, and injects CSS¹⁸ changes using sockets.

¹⁴<https://angular.io>, visited on 03/13/2018

¹⁵MVC ... Model-view-controller

¹⁶MVVM ... Model-View-ViewModel

¹⁷HTML ... Hypertext Markup Language

¹⁸CSS ... Cascading Style Sheets

2. State of the Art

- Nx¹⁹: With nx it is possible to create and build enterprise-grade apps with proven project structure and patterns.
- Universal for ASP.NET: This tool provides an opportunity for building Angular applications in ASP.NET.

The current version is Angular 6, the major release was on April 4, 2018.

With Bootstrap²⁰, responsive and mobile-first sites can be designed very easy. It contains HTML- and CSS-based templates.

¹⁹Nrwl Extensions for Angular

²⁰<http://getbootstrap.com>, visited on 04/02/2018

3. Implementation

In this chapter, the implementation will be described in order to show which infrastructure is used and to explain how the logging, analysing and displaying of the data was handled.

3.1. Infrastructure

Here is a list of all used tools that are used in the context of this project:

- Cloud services platform: Amazon Web Services are used to fulfil these requirements.
- Real time stream processing: Apache Storm is used as real time stream processing tool. The programming language is Java and the integrated development environment IntelliJ IDEA.
- RESTful API: The RESTful API is developed with the integrated development environment Webstorm.
- NoSQL Database: MongoDB, a document database, is used as NoSQL Database.
- Frontend Framework: As well as the RESTful API, the frontend is developed with the integrated development environment Webstorm.

3.1.1. Analytics topology

The analytics topology, which is listed in the first line in Figure 3.1 was created to perform such analyses that are described in this thesis.

In Apache Storm, a so-called “TopologyBuilder” is used to define topologies. An example of running topologies can be seen in Figure 3.1. In this Figure,

3. Implementation

Topology Summary

Name	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)	Scheduler Info
analytics	storm	ACTIVE	10d 1h 13m 2s	1	9	9	1	448	
answer-processing	storm	ACTIVE	24d 3h 29m 12s	3	12	12	1	1344	
evaluation	storm	ACTIVE	48d 4h 16m 48s	2	12	12	0	896	
trigger	storm	ACTIVE	6d 3h 6m 54s	2	32	32	1	896	

Showing 1 to 4 of 4 entries

Figure 3.1.: Example of topologies

all topologies which are used in the context of this project are listed in order to fulfil all real time stream processing requirements.

All spouts of this topology can be found in Figure 3.2. There are the following defined spouts:

- appLogEntrySpout: Handles new messages in the queue that should be processed. The queue contains all logged actions by Android- and iOS app users and is stored by Amazon Simple Queue Service (SQS).
- eventProcessingSpout: This spout is used to check if there are finished football games that should be processed. The status of the game is checked (must be ended) and the test event-flag must be false (which means it is not a game for internally testing). Also, the analytics processed flag must be false, which means that this football game was not processed yet. After this was checked, the bolt is invoked.
- eventRankingSpout: There are two possibilities:
 - Game status is pregame or postgame (logging every five minutes): All games are filtered where the status is pregame or postgame. In this case questions can be answered in the Android- and iOS app and all users in ranking, viewers, anonymous users and visits in ranking are logged every five minutes.
 - Game status is running (logging every minute): When the game is currently running, more precisely has the status first half, first half pause, second half, second half pause, overtime first half, overtime pause, overtime second half or penalty shootout, the ranking is logged every minute.
- followersSpout: Every hour the followers count of all teams and leagues are logged.

3. Implementation

Spouts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error	Error Time
applLogEntrySpout	1	1	250160	250160	230.507	250420	0				
eventProcessingSpout	1	1	120	120	0.000	0	0				
eventRankingSpout	1	1	4900	4900	0.000	0	0				
followersSpout	1	1	600	600	0.000	0	0				
rankingSpout	1	1	500	500	0.000	0	0				

Showing 1 to 5 of 5 entries

Figure 3.2.: Spouts of the analytics topology

- rankingSpout: Every hour all team, league, country and global rankings are logged.

All created bolts are displayed in Figure 3.3. These bolts are defined:

- appLogEntryBolt: In order to log all actions an app user does, all messages that should be processed are handled here. There are following possibilities:
 - NewsViewProcessor: When an app user views a news entry, the news view count will be updated.
 - SponsorClickProcessor: When an app user clicks on a sponsor, the clicks count will be updated.
 - FirehoseProcessor: All logged app actions will be collected and are send to Amazon Kinesis Firehose. They will be send after 50 logged entries, when the user closes the app and when the logout is used.
 - EventVisitsProcessor: When a football game is viewed in the app, the visits count will be incremented.
- logEventBolt: Invoked by the previously explained “eventProcessingSpout”. In this bolt, an event will be analysed and all useful properties will be saved as a document to a created analytics-MongoDB.
- mongoBolt: Used as an interface to write data to the MongoDB which is used in this project.

In order to log specific things, data flows between the defined spouts and

3. Implementation

Bolts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Error Port	Last error	Error Time
appLogEntryBolt	1	1	0	0	0.000	13.435	250400	13.233	250400	0				
logEventBolt	1	1	200	200	0.000	241.400	100	174.250	80	0				
mongoBolt	1	1	0	0	0.000	1.652	5980	1.489	5980	0				

Showing 1 to 3 of 3 entries

Figure 3.3.: Bolts of the analytics topology



Figure 3.4.: Data flow to log clickstreams

bolts are needed. Spouts can invoke bolts and can write to MongoDB. Bolts must be invoked by spouts and cannot start any action independently.

The data flows between spouts and bolts in this project is explained here:

- Log clickstream analysis: After receiving messages from Amazon SQS (appLogEntrySpout), they will be processed in the bolt (appLogEntryBolt). This can be seen in Figure 3.4.
- Log football games: After finding out which games should be processed (eventProcessingSpout), each game will be handled in the logEventBolt and then saved via mongoBolt to the MongoDB. The data flow can be seen in Figure 3.5.
- Log football game rankings: The game ranking is checked every five minutes before and after the game. During each game the ranking is checked every minute. When there was a change in the entry (e.g. more viewers), this entry is saved to the MongoDB, as displayed in



Figure 3.5.: Data flow to log football games

3. Implementation



Figure 3.6.: Data flow to log football game rankings



Figure 3.7.: Data flow to log followers

Figure 3.6.

- Log followers: The count of followers is checked every hour and when it has changed, a new entry is saved to the MongoDB, as shown in Figure 3.7.
- Log rankings: Team-, league-, country-, and global rankings are also checked every hour. When there was a change, a new entry is logged via MongoBolt, as can be seen in Figure 3.8.

An overview about all data flows of these spouts and bolts of the analytics topology, which is used to log data, can be seen in Figure 3.9.

3.1.2. Clickstream user data

In order to perform clickstream analysis, the mobile apps have to log all actions that are described in Chapter Appendix A. When all these actions are logged, there are many possibilities to analyse this data.

After logging 50 entries, when the user closes the app and when the logout is used, this information is transported to the backend via API requests. An Amazon service is used as a web service to guarantee access to message queues that can be processed later. This service is called Amazon Simple



Figure 3.8.: Data flow to log rankings

3. Implementation

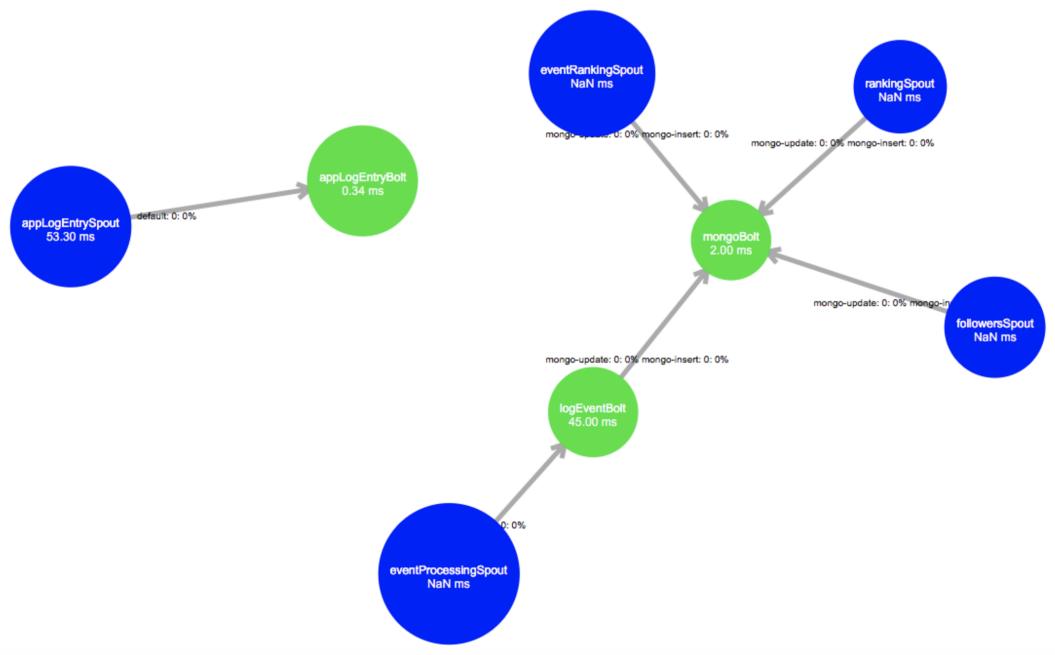


Figure 3.9.: Analytics topology in Apache Storm

3. Implementation

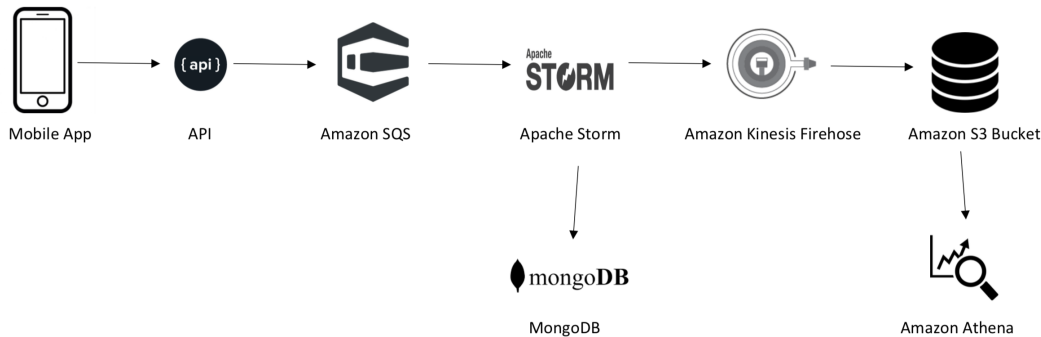


Figure 3.10.: Data Flow in Amazon Athena

Queue Service (Amazon SQS). This stored message is processed in Apache Storm. After receiving such messages in the defined bolt, they will be processed.

On the one hand, the received messages are processed internally. When a news entry is viewed, the news views will be updated in the MongoDB table news-views, when a sponsor click entry is received it will be updated and when a football game is visited in the app, the visits count will be incremented.

On the other hand, the received messages are transported to Amazon Kinesis Firehose, a service for delivering real-time streaming data to various destinations. Amazon Kinesis Firehose collects this data and stores the entries every hour in Amazon S3 buckets which represents the data storage. Afterwards, the app log entries can be queried via Amazon Athena, a query service which can access Amazon S3 buckets directly.

The whole process can be seen in Figure 3.10.

3.2. Log data

A lot of data must be logged to find out the requested information mentioned in previous chapters, e.g. what a moderator has done during a

3. Implementation

football game.

3.2.1. RESTful API

In this subsection, all implementations in Nodejs to log data and other operations in order to achieve this are described.

Promises

With `bluebird`¹, a Nodejs promise library, it is possible to execute asynchronous code in JavaScript. This feature is used in many implemented functions in this project.

Listing 3.1 shows the possibility to add several functions to a list. After all these functions are executed parallel, the return values of all promises can be handled. This method waits with the execution until all promises are done.

```
1 let promises = [];  
2 promises.push(firstAsynchronousFunction());  
3 promises.push(secondAsynchronousFunction());  
4  
5 Promise.all(promises).then(function (returnValues)  
6 {  
7     for (let i = 0; i < returnValues.length; i++) {  
8         // handly return values  
9     }  
10 }).catch(function () {  
11     // handly exceptions  
12 });
```

Listing 3.1: Promises in Nodejs (Promise.all)

¹<http://bluebirdjs.com/docs/getting-started.html>, visited on 03/26/2018

3. Implementation

Another opportunity to implement asynchronous code is stated in Listing 3.2. With this code, functions can be executed sequentially. After the function is done, the return value of this function can be handled.

```
1 return firstAsynchronousFunction().then(function (
    firstReturnValue)
2 {
3     return secondAsynchronousFunction().then(function
        (secondReturnValue)
4     {
5         // handle firstReturnValue and
            secondReturnValue
6     }).catch(function () {
7         // handle exceptions of
            secondAsynchronousFunction
8 });
9
10 }).catch(function () {
11     // handle exceptions of firstAsynchronousFunction
12 });
```

Listing 3.2: Promises in Nodejs (sequential execution)

Moderator activities

In order to find out what a moderator has done before, during and after a football game in the frontend, actions of moderator must be logged. An example of a log activity of a moderator can be found in Listing 3.3, in this case a line-up was entered by a user.

These activities are saved in the events table in the MongoDB. There exists a subdocument for the home and away team and depending on whether the home or away moderator has entered an action, it will be added there.

```
1 {
2     "userid": "57c6be5aff4a2d6e1bff78f4",
3     "action": "add_lineup",
4     "timestamp": 1521014091024
```

3. Implementation

5 }

Listing 3.3: Example of a log activity of a moderator (entering a line-up)

All logged activities of a moderator before, during and after a game are listed here:

- start: When a moderator starts a game.
- start_halftime_pause: When a moderator ends the first half.
- start_second_half: When a moderator starts the second half.
- end_second_half: When there exists the possibility of extra time and a moderator ends the second half.
- start_overtime: When there exists the possibility of extra time and a moderator starts the first half of the overtime.
- end_overtime_first_half: When a moderator ends the first half of the overtime.
- start_overtime_second_half: When a moderator starts the second half of the overtime.
- start_penalty_shootout: When a moderator starts the penalty shootout.
- end: When a moderator blows the final whistle in the frontend.
- delayed_evaluation: When a moderator states that he is present or absent during the game.
- add_activity: When a moderator adds an activity in the game.
All activities: goal, own goal, yellow card, red card, second yellow card, injury, substitution, top chance, top parade, top tackle, offside, penalty goal and penalty miss.
- add_correct_answer: When a moderator adds a correct answer to an evaluable question.
- add_lineup: When a moderator adds a line-up to a team.
- add_question: When a moderator asks a question in the frontend.
- delete_question: When a moderator deletes a question.
- add_post: When a moderator adds a post before, during and after the game.
- delete_post: When a moderator deletes a post.
- add_result: When a moderator enters the final score of the first half, second half, overtime first half, overtime second half, penalty shootout and final score.

3. Implementation

3.2.2. Clickstream user data

In order to make reliable statements, it would not be enough to save only count of clicks of actions of a user, because then a lot of information will be lost. In this case one cannot query how much Android and how much iOS user do certain actions. It is more useful to log the timestamp of the occurrence and a lot of other user data to guarantee a meaningful statement out of each entry.

The challenge was to write down all useful actions that should be logged by the Android- and iOS apps. It was clicked through each screen of the app and all things that should be logged were written down.

In order to answer all relevant questions, the following information must be logged:

- action: All possible actions a user can do (open, close, login, logout, register, rate, view, click, share).
- category: Either general app things (app, notification, deep link) or collections in the database (such as news, events, questions, posts and so on) are logged.
- type: Contains an optional description of the category.
- id: Contains an optional id, when a logged action corresponds with a collection from the MongoDB (e.g. a specific news article is viewed, then the newsid of this article is stored in the database).
- userid: Contains the identification number of the user who made an action.
- sessionid: In order to save a token to identify a session of a user.
- useragent: In order to identify which app version and which operating system is used (e.g. 2.5.14; Android/API 25).
- timestamp: Contains a Unix timestamp of the occurred action.
- partitiondatetime: Is used for Amazon Kinesis Firehose, because then not all data must be searched every time. In this case all specific actions during a specific time range can be queried.

All logged actions by the Android- and iOS apps ordered by relevant categories can be found in Chapter Appendix [A](#).

3. Implementation

An example of such a log entry can be seen in Listing 3.4. In this case an app user has clicked on a news article in the news screen via an Android smartphone and has used the app version 2.5.19 on 03/17/2018 between 9 and 10 am.

```
1  {
2    "action": "click",
3    "category": "news"
4    "type": "newsscreen",
5    "id": "580731b87349222878c01122",
6    "userid": "5a86a9c4b884ae5d2b60528d",
7    "sessionid": "0be89849-8ba2-4eb6-97a8-9e18658b4889",
8    "useragent": "2.5.19; Android/API 24",
9    "timestamp": 1521277837162,
10   "partitiondatetime": "2018-03-17-09",
11 }
```

Listing 3.4: Example of a log entry

News views

There exists a MongoDB table with the name news_views, an example entry can be seen in Listing 3.5, where all users are logged which have viewed a news article. With the logging of this information it is easy to find out how many different users has viewed a specific news article (identified by newsid).

```
1  {
2    "_id" : ObjectId("5a3be9865d0daa92a10bd659"),
3    "newsid" : ObjectId("5a3bde6c5bf45a59274899bd"),
4    "viewers" : [
5      ObjectId("59e46fa991c8853457054cea"),
6      ObjectId("5a63aa1dd23b4410a126bebc"),
7      ObjectId("59d9fdd0cfc4b92ef031302c"),
8      ObjectId("58be780114deb6216156ecdb"),
9      ObjectId("58c3b953d00afb511cff567b"),
```

3. Implementation

```
10     ObjectId("5a6c6cefce8f147d5621845f"),
11     ObjectId("58c30d88bce4673e99a5c62b"),
12     ObjectId("58c036ece6eb18789a02b86c"),
13     ObjectId("58fa0a96abf0b958f1fc800a")
14 ]
15 }
```

Listing 3.5: News views

Sponsor clicks

There exists also a MongoDB table with the name `sponsor_clicks`, an entry is displayed in Listing 3.6, where all users are saved which have clicked on a sponsor. It follows the same schema as the viewed news articles mentioned before.

```
1 {
2   "_id" : ObjectId("5a75f9f3fe532d17d9e49e8e"),
3   "sponsorid" : ObjectId("59006c8889e2b267e004656e"),
4   "users" : [
5     ObjectId("59c60271141b7d0ecbed74e2")
6   ]
7 }
```

Listing 3.6: Sponsor clicks

3.2.3. Time series

As explained in MongoDB (2018[a]), a time series is a sequence of data points saved over a time interval and is used in many areas like the financial market, systems (server logs) and sensors (temperatures). When time series are displayed in charts, current trends of this data can be analysed. Time series are very useful to make historical and predictive analysis of different information.

As can be seen in Box et al. (2015), there are many usages of time series. A very huge topic is the forecasting of future values of time series. Another

3. Implementation

possibility is the examination of relationships between several related time series variables of interest.

Game Ranking

Time series are used to save rankings. The ranking is checked every 5 minutes before and after a game and during a game the ranking is checked every minute. When something has changed, a new entry will be saved. An example entry of logged time series of a game ranking can be found in Listing 3.7.

```
1 {
2   "_id" : ObjectId("5a9cf9889580d11422a03368"),
3   "eventid" : ObjectId("5a99129d6008d9677060a708"),
4   "teams" : {
5     "home" : {
6       "teamid" : ObjectId("586169d1de03c06083281083"),
7       "entries" : [
8         {
9           "viewers" : 22,
10          "users" : 9,
11          "anonymous_users" : 0,
12          "visits" : 5,
13          "timestamp" : 1520236935864
14        },
15        {
16          "viewers" : 23,
17          "users" : 10,
18          "anonymous_users" : 0,
19          "visits" : 6,
20          "timestamp" : 1520237535960
21        },
22        ...
23        {
24          "viewers" : 41,
25          "users" : 22,
```

3. Implementation

```
26         "anonymous_users" : 1,
27         "visits" : 50,
28         "timestamp" : 1520269338083
29     }
30 ]
31 },
32 "away" : {
33     "teamid" : ObjectId("586169d3de03c06083281130"),
34     "entries" : [
35         {
36             "viewers" : 0,
37             "users" : 0,
38             "anonymous_users" : 0,
39             "visits" : 5,
40             "timestamp" : 1520236935864
41         },
42         ...
43         {
44             "viewers" : 0,
45             "users" : 0,
46             "anonymous_users" : 0,
47             "visits" : 40,
48             "timestamp" : 1520267537994
49         }
50     ]
51 }
52 }
```

Listing 3.7: Log entry of event ranking

Rankings

All team, league, country and global rankings are checked every hour by a spout and when it has been changed, the new entry will be logged. In this case all users in ranking and anonymous users are logged. Such an entry is displayed in Listing 3.8.

3. Implementation

```
1 {
2   "_id" : ObjectId("5a12f01b9580d1361a1deb6f"),
3   "leagueid" : ObjectId("586137dfde03c06083280e79"),
4   "entries" : [
5     {
6       "timestamp" : 1511190555916,
7       "users" : 36,
8       "anonymous_users" : 10
9     },
10    {
11      "timestamp" : 1513005116794,
12      "users" : 37,
13      "anonymous_users" : 10
14    },
15    ...
16  ]
17 }
```

Listing 3.8: Log entry of league followers

Followers

Also, all team and league followers are logged every hour, when the follower count has changed, as can be seen in Listing 3.9. With this information it is very easy to detect less motivated moderators of teams and a predictive analysis can be made.

```
1 {
2   "_id" : ObjectId("5a0b4e9d0b5a4674cce9e115"),
3   "teamid" : ObjectId("586169d6099e0d6089bdb3b6"),
4   "entries" : [
5     {
6       "timestamp" : 1510690461774,
7       "followers" : 190
8     },
9     {
10      "timestamp" : 1511716299259,
```


3. Implementation

```
11     "followers" : 191
12   },
13   {
14     "timestamp" : 1511987954182 ,
15     "followers" : 192
16   },
17   {
18     "timestamp" : 1512486233801 ,
19     "followers" : 193
20   },
21   {
22     "timestamp" : 1519545576794 ,
23     "followers" : 194
24   }
25 ]
26 }
```

Listing 3.9: Log entry of team followers

3.2.4. Month rankings

In order to answer the question which users has collected the most points in the last month, month rankings of all teams, leagues, states and countries are logged.

After a question has been answered by a user, the month ranking will be updated.

When a month has finished, the whole month ranking will be saved to the analytics-MongoDB. There exists a month ranking for teams, leagues, states and countries.

3.2.5. Game specific details

Game specific details must be logged to answer questions in relation to a football game. With this information, statements about moderator activities

3. Implementation

and users who follow the game can be made.

Here is an explanation of the most important properties of a game in the MongoDB:

- `teams.home.abbreviation`, `teams.away.abbreviation`: Contains the abbreviation of the team name which is displayed in the apps.
- `teams.home.moderation_type`, `teams.away.moderation_type`: Indicates which users moderate the match.
 - `standard`: There is no moderator and there will be only automated questions and posts.
 - `team`: There is a club moderator.
 - `fan`: There is a fan moderator.
 - `redactional`: This game will be redactional moderated by a neutral viewer.
- `teams.home.lineup`, `teams.away.lineup`: Home and away indicate the home and away team. This property contains all players from the line-up with `playerid` (there exists a Mongo table called `players`), percentage of the fans who have voted this player in their line-up and the position of the player (goalkeeper, defence, midfield, offense).
- `teams.home.reserve`, `teams.away.reserve`: Contains all reserve players.
- `teams.home.formation`, `teams.away.formation`: Indicates the used formation (e.g. 3-4-3 or 4-4-2) of a team.
- `teams.home.live_lineup`, `teams.away.live_lineup`: Contains the current line-up on the pitch (line-up, reserve and replaced players).
- `teams.home.subscribers`, `teams.away.subscribers`: The subscribers are all users who receive notifications when one of the following actions occur:
 - `questions`: When a new question is asked.
 - `news`: When a new news article appears.
 - `goals`: When the home or away team has scored.
 - `interevent_questions`: When not game specific questions are asked.
 - `event_posts`: When a new post emerges in a game.
 - `general_posts`: When not game specific posts appear.
- `teams.home.viewers`, `teams.away.viewers`: The home and away viewers of the game in the app.

3. Implementation

- actions: The action property contains logged actions from the moderator of a football game. With this data, it can be determined how active a moderator was before, during and after the game.
These actions include start, start halftime pause, start second half, end second half, start overtime, end overtime first half, start overtime second half, start penalty shootout, end, delayed evaluation, add activity, add correct answer, add lineup, add question, delete question, add post, delete post, add result and were already explained in Section 3.2.1.
- timestamps: Contains timestamps of a specific game. These timestamps are Unix timestamps with 13 digits. There exist the following possibilities:
 - pregame.start: Means the moment when questions and posts of the game appear in the app.
 - postgame.end: Means the moment when no questions and posts can be written by a moderator and users cannot answer questions of a specific game anymore.
 - start: Contains the official kick-off of the game.
 - end: Contains the timestamp of the end of the game.
 - regular.firsthalf.start: Contains the timestamp of the start of the first half. When the game has not started yet, this property is null.
 - regular.firsthalf.end: Contains the timestamp of the end of the first half.
 - regular.secondhalf.start: Contains the timestamp of the start of the second half.
 - regular.secondhalf.end: Contains the timestamp of the end of the first half.
 - overtime.firsthalf.start: Contains the timestamp of the start of the first half of the overtime.
 - overtime.firsthalf.end: Contains the timestamp of the end of the first half of the overtime.
 - overtime.secondhalf.start: Contains the timestamp of the start of the second half of the overtime.
 - overtime.secondhalf.end: Contains the timestamp of the end of the second half of the overtime.
 - overtime.shootout.start: Contains the timestamp of the start of

3. Implementation

the penalty shootout.

- scores: Here all results after the first half, second half, overtime first half, overtime second half and penalty shootout are saved.
- leagueid: Contains the identification number of the league in the MongoDB. With this property, the name of the league, the state name and the country name can be determined.
- analytics_processed: When this property is false, the event was not processed yet.
- test_event: When the game is only a test match, this property is true.
- userid: Contains the identification number of the user who has created this football game in the frontend.
- _created: This property contains a timestamp which marks the creation of the game.
- _modified: Contains a timestamp of the game, when the last modification of this game was made.
- visits: Contains all visits through the Android and iOS-App.

There is a check every ten minutes, as can be seen in previous mentioned spout (Section 3.1.1), if the analytics_processed-flag is not set, the game status is ended, and it is not a test match. In this case, the game will be processed, and all needed information will be exported.

First of all, the basic information of a football game will be exported like the event-id, the two team-ids, all timestamps, the league-id, all actions the moderator has done (adding activities, adding correct answers, his setting of present or absent, the entered line-up). With this information it can be said what a moderator has entered.

Also, other things like an edited sponsor, if eleven players for the line-up has been entered, how many questions have been asked, how many posts have been entered and how many news entries were written in the last two weeks are logged.

The most important properties and how they are saved are here explained:

- teams.home.moderators, teams.away.moderators: All moderators who have done anything during the game or have written a news article in the last seven days are logged here.

3. Implementation

- teams.home.questions, teams.away.questions: Here the count of all evaluated questions and evaluable questions, of all automated and evaluable automated questions, of all manual and evaluable questions and if the line-up was entered are saved.
- teams.home.posts, teams.away.posts: Contains the count of automated posts with and without pictures, the count of manual entered posts with and without pictures before, during and after the game.
- teams.home.activities.home, teams.home.activities.away: Here are all activities from the home and away team stored and how many have been entered by the moderator (goal, own goal, yellow card, red card, yellow-red card, injury, substitution, top chance, top tackle, top parade, offside, penalty goal, penalty miss).
- teams.home.news, teams.away.news: Contains all news articles with timestamp, title, date and username (of the moderator who has written this article) of the last seven days.
- actions: Contains all actions of moderators like adding activities (goals, own goals etc.), adding correct answers to evaluable questions, delayed evaluation (with this flag the moderator of a game can decide if he is present or absent during the game), add line-up, kick-off and end timestamps of the match. In all these cases, the team (home or away), username, timestamp and further interesting information are saved.

3.3. Analyse data

The logged data, which was described in the previous Chapter, must be processed and analysed. It is not enough to log this data, because all this information must be displayed.

3.3.1. RESTful API

There are several API routes for getting the logged information via API requests to fulfil the requirements of this thesis.

3. Implementation

Statistics

In order to deliver statistics about the clicks on news articles and sponsor hyperlinks, API routes were written.

One GET-request to deliver the count of a specific viewed news article (/statistics/news/:id) and another GET-request for the count of clicks on a specific sponsor (/statistics/sponsor/:id).

For the news information, a JSON Object consisting of the newsid and a views count is delivered, for the sponsor information a JSON Object with the sponsorid and sponsor clicks.

These statistics can also be extended, when another information is needed or claimed by the customer.

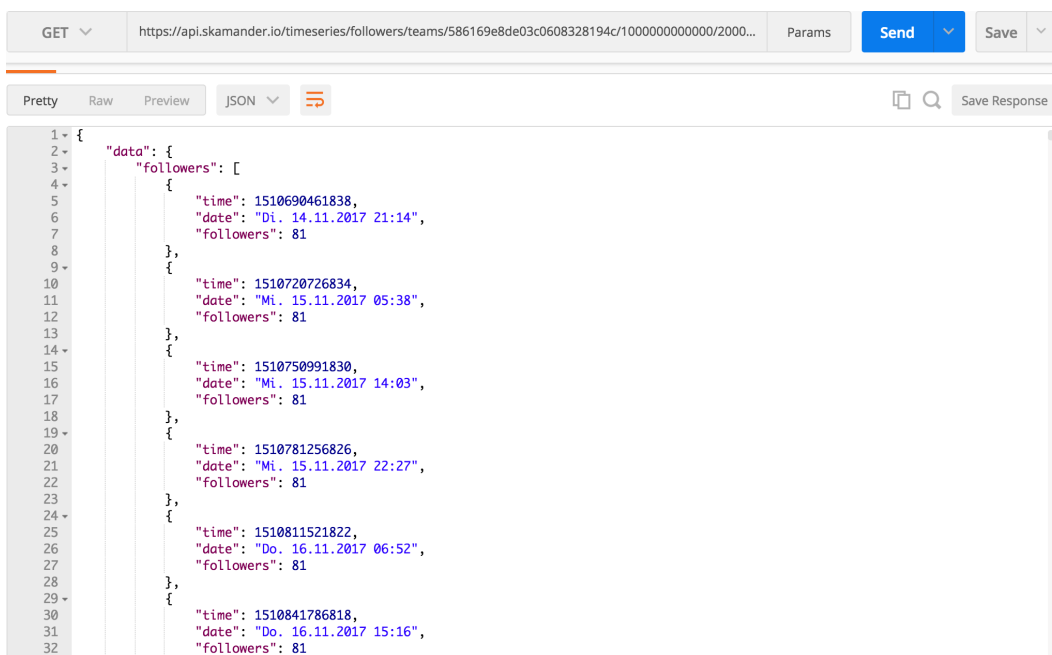
Time series

After logging time series of rankings and followers, time series must be calculated. 360 entries are computed in order to receive meaningful time series. With these entries historical and predictive analysis can be made.

Following routes are provided to get time series:

- Game ranking: The identification number of a game must be provided (GET request: /timeseries/ranking/events/:id).
- Team and league ranking: To receive a team or league ranking, a value for collection (team or league), the teamid or leagueid and the start and end timestamp must be provided (GET request: /timeseries/ranking/:collection/:id/:start/:end).
- Global ranking: In order to receive the global ranking over all leagues, states and countries, a start and end timestamp must be provided (GET request: /timeseries/ranking/global/:start/:end).
- Followers: In order to get the followers of a team or a league, the collection must be provided, as well as the teamid or leagueid and a start and end timestamp (GET request: /timeseries/followers/:collection/:id/:start/:end). The response of a GET request to receive all followers within a time range of a specific team can be seen in Figure 3.11.

3. Implementation



The screenshot shows a REST client interface with a GET request to the URL `https://api.skamander.io/timeseries/followers/teams/586169e8de03c0608328194c/100000000000/2000...`. The response is a JSON object with a `data` property containing an array of follower records. Each record includes a `time` (Unix timestamp), a `date` (human-readable date and time), and a `followers` count of 81.

```
1- {
2-   "data": {
3-     "followers": [
4-       {
5-         "time": 1510690461838,
6-         "date": "Di. 14.11.2017 21:14",
7-         "followers": 81
8-       },
9-       {
10-        "time": 1510720726834,
11-        "date": "Mi. 15.11.2017 05:38",
12-        "followers": 81
13-      },
14-      {
15-        "time": 1510750991830,
16-        "date": "Mi. 15.11.2017 14:03",
17-        "followers": 81
18-      },
19-      {
20-        "time": 1510781256826,
21-        "date": "Mi. 15.11.2017 22:27",
22-        "followers": 81
23-      },
24-      {
25-        "time": 1510811521822,
26-        "date": "Do. 16.11.2017 06:52",
27-        "followers": 81
28-      },
29-      {
30-        "time": 1510841786818,
31-        "date": "Do. 16.11.2017 15:16",
32-        "followers": 81
33-      }
34-    ]
35-   }
36- }
```

Figure 3.11.: GET request for receiving followers within a time range

3. Implementation

3.3.2. Serverless Interactive Query Service

Amazon Athena, which was described in Section 2.2, is used as a serverless interactive query service. With Amazon Athena it is possible to query Amazon S3 buckets directly, where all logged app actions of the Android- and iOS users are collected.

An example query can be seen in Listing 3.10, where the overall news view count since a specific date is requested. In this query the user agent is parsed after a blank and before a slash, the return value will be Android or iOS.

A user agent can look like:

- For Android: 2.5.16; Android/API 24
- For iOS: 1.2.1 iOS/11.2.6

Then the logged app actions are queried, where a news article is viewed.

There are three possibilities to achieve this:

- When an app user clicks on a notification to view a new news article.
- When an app user clicks on a deep link to view a new news article.
- When a news article is viewed via the news screen.

Then the count of Android and iOS viewers of a specific news article is calculated.

```
1  SELECT count(*) as count ,
2  split(substr(useragent, strpos(useragent, ' '),
3  strpos(useragent, '/')), '/') [1]
4  as os, id FROM default.logdata
5  WHERE partitiondatetime >= '2018-03-20-00' AND
6  ((action = 'click' AND
7  category = 'deeplink' AND
8  type = 'news')
9  OR
10 (action = 'click' AND
11 category = 'notification' AND
12 type = 'news')
13 OR
14 (action = 'open' AND
```


3. Implementation

```
15     category = 'news')) GROUP BY
16     id, split(substr(useragent, strpos(useragent, ' '),
17     ),
    strpos(useragent, '/'), '/') [1]
```

Listing 3.10: Amazon Athena query to get all Android, iOS and overall news view counts since a specific date

The output of this query is displayed in Table 3.1.

count	operating system	id
5	Android	5ab03671a86960035dd7df78
1	Android	5aac05876f445e4aofb25654
1	Android	59e45ebeg91c8853457054ce8
4	Android	5a5527904acb750b7771efab
1	Android	59ad2eod6687ce1599obb066
1	iOS	5aa70e9a8c9b7c7a601b0cb5

Table 3.1.: Output of an Amazon Athena query to receive the Android and iOS viewed news articles count

Amazon Athena also has a graphical user interface from Amazon to test queries online, it can be seen in Figure 3.12.

3.3.3. Social media analysis

For social media analysis, Facebook provides the Graph API² in order to receive information about Facebook pages.

With this API it is possible to get information of public accessible Facebook pages. For private profiles, Facebook must be contacted, and it must be shown for which purpose this information is needed.

Fb³ is a module, developed for Node.js, to use Graph API and was used in this context to get the described information.

²<https://developers.facebook.com/docs/graph-api/>, visited on 04/01/2018

³<https://www.npmjs.com/package/fb>, visited on 04/01/2018

3. Implementation

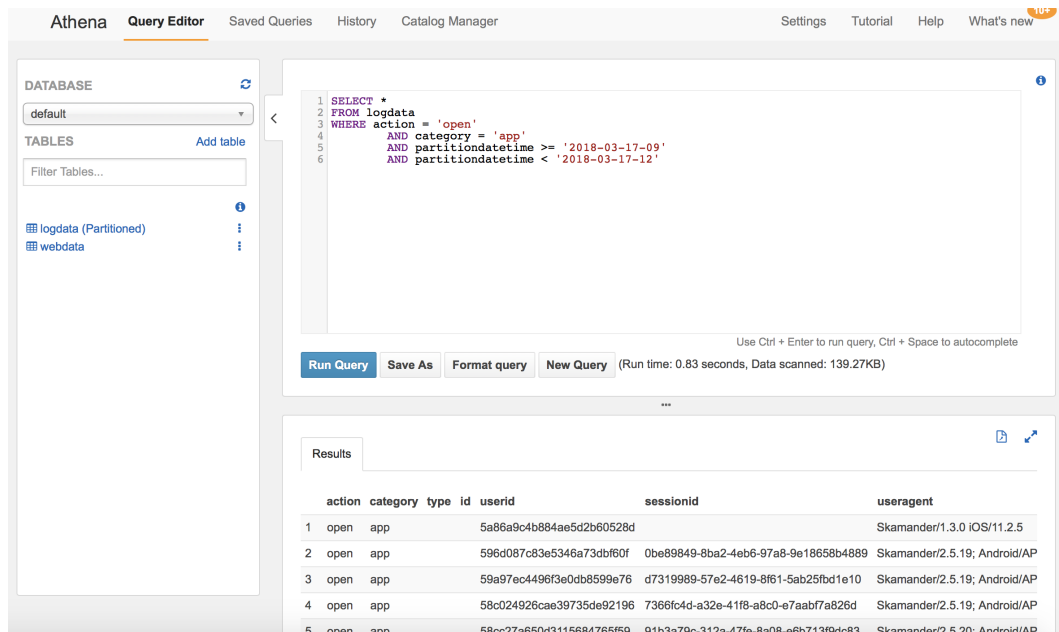


Figure 3.12.: Example query in the GUI of Amazon Athena

Facebook feed

With the provided code in Listing 3.11, it is possible to get information about the last posts of this Facebook page.

```
1 facebook.api('', 'post', {
2     batch: [
3         {
4             method: 'get',
5             relative_url: team + '/feed?fields=created_time,
6                 message,link,attachments,permalink_url,' +
7                 'shares,reactions.type(LIKE).limit(0).summary(1).as(
8                 like),reactions.type(LOVE).limit(0).summary(1).as(
9                 love),' +
10                'reactions.type(HAHA).limit(0).summary(1).as(haha),
11                reactions.type(WOW).limit(0).summary(1).as(wow),'
12                +
13                'reactions.type(SAD).limit(0).summary(1).as(sad),
14                reactions.type(ANGRY).limit(0).summary(1).as(angry
15                )' +
16                '&limit=100'
```

3. Implementation

```
10     },
11     ]
12   }, function(posts) {
13     let response = JSON.parse(posts[0].body);
14   }
15 );
```

Listing 3.11: Gain Facebook timeline of a public accessible page

One Facebook post contains:

- `created_time`: Contains the date of the written post.
- `message`: Contains the written text (when existing).
- `link`: Contains the shared link (when existing).
- `attachments`: Contains the shared attachment in this posting (when existing). In most cases the attachment is a preview picture of the shared link.
- `permalink_url`: Contains a hyperlink to the shared posting of the team which shared it on Facebook.
- `shares`: How many people shared this Facebook post?
- `like`: How many people liked this Facebook post?
- `love`: How many people reacted to this Facebook post with love?
- `haha`: How many people reacted to this Facebook post with haha?
- `wow`: How many people reacted to this Facebook post with wow?
- `sad`: How many people reacted to this Facebook post with sad?
- `angry`: How many people reacted to this Facebook post with angry?

The limit specification means the count of Facebook posts that should be returned.

Figure 3.13 shows the response of such a request to get all Facebook posts written by the team “SV Lafnitz” from the Austrian “Regionalliga Mitte”.

Another helpful module is `URL`⁴, a module for Node.js applications, which is used for URL resolution and parsing.

It is needed to parse the hyperlinks shared on Facebook by teams, as can be seen in Listing 3.12. The `pathname` contains the remaining characters after a domain ending. With this listing it is possible to classify each link and

⁴<https://www.npmjs.com/package/url>, visited on 04/01/2018

3. Implementation

determine if it contains a shared event, ranking, news, line-up, question, post, fixtures or another information.

```
1 let facebook_url = new URL(post.link);
2 let facebook_pathname = facebook_url.pathname;
3
4 if (facebook_pathname.indexOf("/event/") !== -1) {
5   ret.link.event++;
6 } else if (facebook_pathname.indexOf("/ranking/") !== -1) {
7   ret.link.ranking++;
8 } else if (facebook_pathname.indexOf("/news/") !== -1) {
9   ret.link.news++;
10 } else if (facebook_pathname.indexOf("/lineup/") !== -1) {
11   ret.link.lineup++;
12 } else if (facebook_pathname.indexOf("/question/") !== -1) {
13   ret.link.question++;
14 } else if (facebook_pathname.indexOf("/post/") !== -1) {
15   ret.link.post++;
16 } else if (facebook_pathname.indexOf("/fixtures/") !== -1) {
17   ret.link.fixtures++;
18 } else {
19   ret.link.other++;
20 }
```

Listing 3.12: Classify shared hyperlinks on Facebook

Facebook fan count

The fan count of a Facebook page is needed to evaluate the importance of a team on this platform. It works similar to getting all posts of a team. The request can be seen in Listing 3.13.

```
1 facebook.api('', 'post', {
2   batch: [
3     {
4       method: 'get',
5       relative_url: team + '?fields=fan_count'
6     },
7   ]
8 }, function (response) {
9   let fan_count = JSON.parse(posts[0].body.fan_count);
10 }
```

3. Implementation

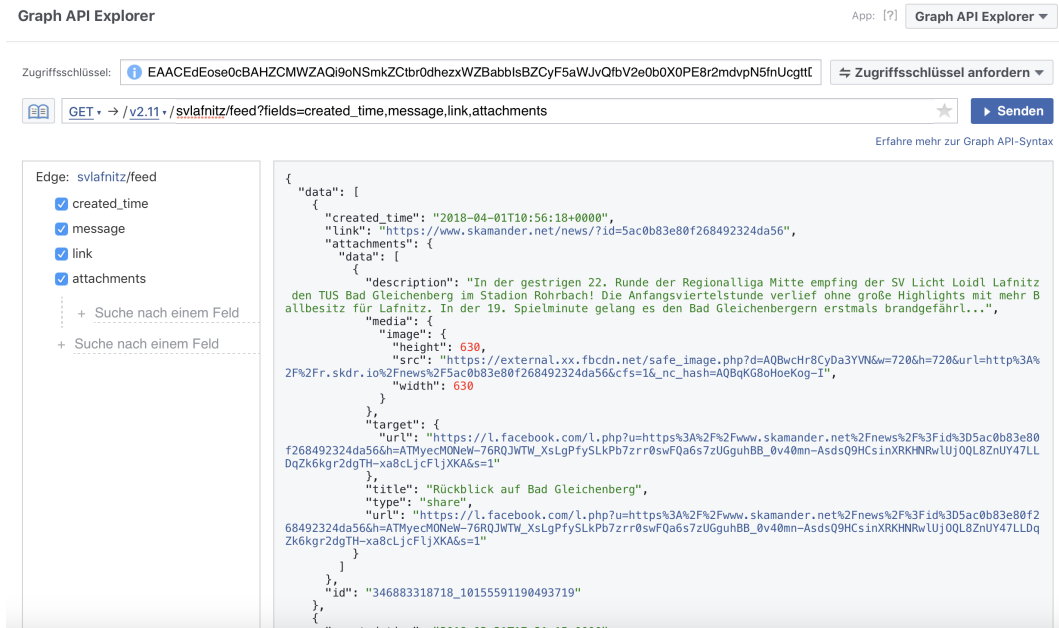


Figure 3.13.: Gain Facebook timeline of a team via Graph API

11);

Listing 3.13: Gain Facebook fan count of a public accessible page

Figure 3.14 shows the response of this request. In this example, the likes count of the team “SV Lafnitz” on Facebook is shown.

3.3.4. Moderator statistics

An algorithm was implemented to discover how active moderators were during a football game.

The most important things:

- Has the moderator started the first half, ended the first half, started the second half, ended the second half?
- Has the moderator entered the line-up?

3. Implementation



Figure 3.14.: Gain Facebook fan count of a team via Graph API

- Has the moderator of a football game also entered other activities like top parade, top chance, etc. beside goals?
- Has the moderator entered some game comments during a game?
- Has the moderator evaluated all questions?

After this was analysed, each moderation of a team in a game was classified in weak, average and strong moderation.

3.3.5. Game statistics

In order to determine game specific statistics, games were analysed to receive the following information:

- most used formations at wins, draws, losses at home, away and both together
- scored home and away goals
- conceded home and away goals

3.4. Display data

After logging and analysing the gathered information, the data must be displayed to show the outcome of the work of this thesis.

3. Implementation

3.4.1. Internal communication

Not all information is intended for the general public or it will cost something to receive specific information. In this project, Google Drive⁵ is used for sharing development hints and project intern communication.

Google Drive integration

The reports with the information about teams, leagues, states and countries are published via Google Drive.

Google Drive API⁶ allows users to make basic operations in Google Drive.

Basic operations in Google Drive are listed here:

- Copy files: With a specific file id, a copy of a Google Drive file can be created.
- Create files: It is possible to create a file in Google Drive. The mime type of the file indicates the file type.
- Delete files: With a specific file id, a file can be deleted permanently without moving it to trash.
- Get file: With this operation, metadata and content of a file can be determined.
- Search for files: It is possible to search for files via Google Drive API. Listing 3.14 shows a search for spreadsheet files with a specific name. The id of the first file, that was found, is returned.

```
1 service.files.list({
2   auth: auth,
3   q: "mimeType = 'application/vnd.google-apps.
      spreadsheet' and name='" + name + "'",
4 }, function (err, response) {
5   if (err || response.files.length === 0) {
6     console.log('File does not exist');
7     reject(err);
8   }
9   } else {
```

⁵<https://www.google.com/drive/>, visited on 04/01/2018

⁶<https://developers.google.com/drive/>, visited on 04/01/2018

3. Implementation

```
10         fulfill(response.files[0].id);
11     }
```

Listing 3.14: Get specific file id in Google Drive

- Update files: One possibility to update a file is to move a file between folders.

Firstly, the existing parents of the file will be retrieved via file id and after then the file will be moved to a specific folder via folder id. This can be seen in Listing 3.15.

```
1  service.files.get({
2      auth: auth,
3      fileId: fileId,
4      fields: 'parents'
5  }, function (err, file) {
6
7      // Move the file to the new folder
8      let previousParents = file.parents.join(',');
9      service.files.update({
10         fileId: fileId,
11         addParents: folderId,
12         removeParents: previousParents,
13         fields: 'id, parents',
14         auth: auth
15     }, function (err, file) {
16         if (err) {
17             console.error(err);
18         } else {
19             cb(url);
20         }
21     });
22 });
```

Listing 3.15: Move file to a specific folder in Google Drive

Generation of Google Sheets

After basic Google Drive operations are made, the information is shared via Google Spreadsheets⁷. Google Spreadsheets are very useful, because there it is possible to filter specific rows and information.

⁷<https://developers.google.com/sheets/>, visited on 04/01/2018

3. Implementation

Google Sheets API⁸ provides a lot of possibilities to make basic operations and format a Google Spreadsheet.

Basic Google Spreadsheet operations include:

- Create spreadsheets: For the creation of a spreadsheet, a title, tab title and an authentication are needed. It is created at the basic folder of Google Drive. After the creation, a file can be moved to a specific folder via Google Drive API, as described earlier.
- Update values of spreadsheets: With a given spreadsheet id, values and formatting information (e.g. value input option must be “RAW” when the spreadsheet contains formulas) of a specific spreadsheet can be updated.
- Batch update: With this operation, one or more requests can be applied to the spreadsheet. Before all operations are executed, the operations are checked. After applying all requests, some of them can contain replies with important information for further use. An example for this is when a tab is added to a spreadsheet, the response of this operation contains the new sheet id of this tab.
- Get spreadsheets: With a given spreadsheet id, information about this spreadsheet can be gathered. The response contains information about the title, basic formatting, the contained sheets and their formatting and URL of the spreadsheet.

Following basic sheet operations are available:

- Add sheets: Sheets API provides the possibility to add tabs in a Google Spreadsheet on a specific position (specified by index, the value 0 indicates the first position).
- Delete sheets: With a given sheet id, a tab can be deleted. When there is no tab with such an id or it is the last tab in the whole Google Spreadsheet, an exception occurs. A Google Spreadsheet must contain at least one sheet.

With Google Sheets API, the following operations for formatting are possible, as can be seen in Google (2018), and used in this project:

⁸<https://developers.google.com/sheets/api/>, visited on 04/01/2018

3. Implementation

- User entered format: With this operation, a specific cell can be updated. There are the following possibilities:
 - Background colour: This colour is given in red, green and blue values (e.g. red: 1.0, green: 1.0, blue: 0 represents yellow).
 - Text format: Foreground colour, font size, font family, underlining, strikethrough, italic and thickness can be edited (e.g. `fontSize: 11`, `bold: true`).
 - Horizontal alignment: The possibilities are left, center and right.
 - Vertical alignment: The opportunities are top, middle and bottom.
 - Wrap strategy: There are several ways to define the wrap strategy:
 - * `Wrap`: Lines that are longer than the cell will be wrapped and not clipped.
 - * `Clip`: When lines are longer than the cell, the word will be clipped.
 - * `Overflow_cell`: When lines are longer than the cell, the word will be continued in the next cell unless in the next cell is also text, then it is the same command as clip.
 - Number format: The format of the cell can be specified. In order to sort the date in Google Spreadsheet, the type date is used with the pattern "dd.mm.yyyy hh:mm". Other possibilities are percent, currency, date, number and text.
 - Borders: Top, left, right and bottom borders can be defined.
 - Padding: Top, left, right and bottom paddings can be defined.
- Update dimension properties: With this property, the height (if a row) or width (if a column) of the cell can be defined in pixels.
- Auto resize dimension: Based on the content of the cells, they will be resized to see all needed information.
- Delete dimension: With this operation, unused columns and rows can be deleted.
- Merge cells: Cells can be merged in Google Spreadsheet. Either all given cells in a range are merged (with `MERGE_ALL`) or only the columns and not the rows are merged (with `MERGE_COLUMNS`).
- Update values: There exists two types to describe the data. When the Google Spreadsheet only contains raw data or when formulas are used, the types `RAW` and `USER_ENTERED` must be defined.
- Frozen rows: This property provides the possibility to freeze one or

3. Implementation

more rows of a Google Spreadsheet.

- Frozen columns: This property provides the possibility to freeze one or more columns of a Google Spreadsheet.
- Basic filter: With this operation, a basic filter can be enabled to sort the content of each column. In order to sort dates, the number format of the column must be changed to the type date and a specified pattern.

There are also a lot of other possibilities to work with Google Spreadsheets like specifying and embedding a chart, but not used in this project.

Writing Emails

Beside the possibility of generating Google Spreadsheets and adding it to a specific Google Drive folder, there are also other ways to publish information like writing emails via Node.js.

Nodemailer⁹ is a module for Node.js applications for sending emails. An implementation can be seen in Listing 3.16.

“sendEmail” represents the function call for sending emails. After sending an email, the function returns to the callback and then the process will be finished.

```
1 IEmail.sendEmail(  
2   "Admin",  
3   "admin@gmail.com",  
4   mail_addresses,  
5   subject,  
6   "HTML only...",  
7   html,  
8   replyto,  
9   error => {  
10    process.exit()  
11  }  
12 );  
13  
14 let nodemailer = require('nodemailer');  
15 let smtpTransport = require('nodemailer-smtp-transport');
```

⁹<https://nodemailer.com/about/>, visited on 04/01/2018

3. Implementation

```
16 let transport = nodemailer.createTransport(smtpTransport(config
    .aws_ses));
17
18 function sendEmail (from_name, from_address, to_addresses,
    subject, plaintext, htmltext, replyto, cb)
19 {
20
21     var mailOptions = {
22         from: from_name + ' <' + from_address + '>', // sender
            address
23         to: to_addresses, // list of receivers e.g: 'test@test.at,
            test12@test.at'
24         subject: subject, // Subject line
25         text: stripHtmlTags(htmltext), // plaintext body
26         html: htmltext // html body
27     };
28
29     if(replyto !== undefined && replyto !== null)
30         mailOptions.replyTo = replyto;
31
32     transport.sendMail(mailOptions, function(error) {
33         if (error) {
34             console.log(error);
35             if (cb) {
36                 cb(error);
37             }
38         }
39         else {
40             console.log("Email successfully sent");
41             if (cb !== undefined) {
42                 cb();
43             }
44         }
45     });
46 }
```

Listing 3.16: Generating emails

3. Implementation

Writing Slack messages

There is also another opportunity to publish information, because in this project, Slack¹⁰ is used for intern project communication.

Slacknode¹¹ is a module for Node.js applications to send messages in Slack. An implementation can be seen in Listing 3.17.

```
1 let slackNode = require('slack-node');
2 let slack = new slackNode(config.slack_api_key);
3
4 slack.api('chat.postMessage', {
5   text: slack_message,
6   channel: '#statistiken',
7   username: config.environment + "-statistiken"
8 }, (err, response) => {
9   if (err) {
10    console.err(err);
11   }
12   process.exit();
13 }
14 );
```

Listing 3.17: Generating slack messages

This service is used, when a new report is updated. Figure 3.15 shows written slack messages after several updates of different reports.

3.4.2. External communication

There exists already a so-called administration page, newly developed in Angular 5, where the logged and analysed data will be presented.

In this web application, information which is disseminated to the general public is published like time series of followers counts.

¹⁰<https://https://slack.com/>, visited on 04/01/2018

¹¹<https://www.npmjs.com/package/slack-node>, visited on 04/01/2018

3. Implementation

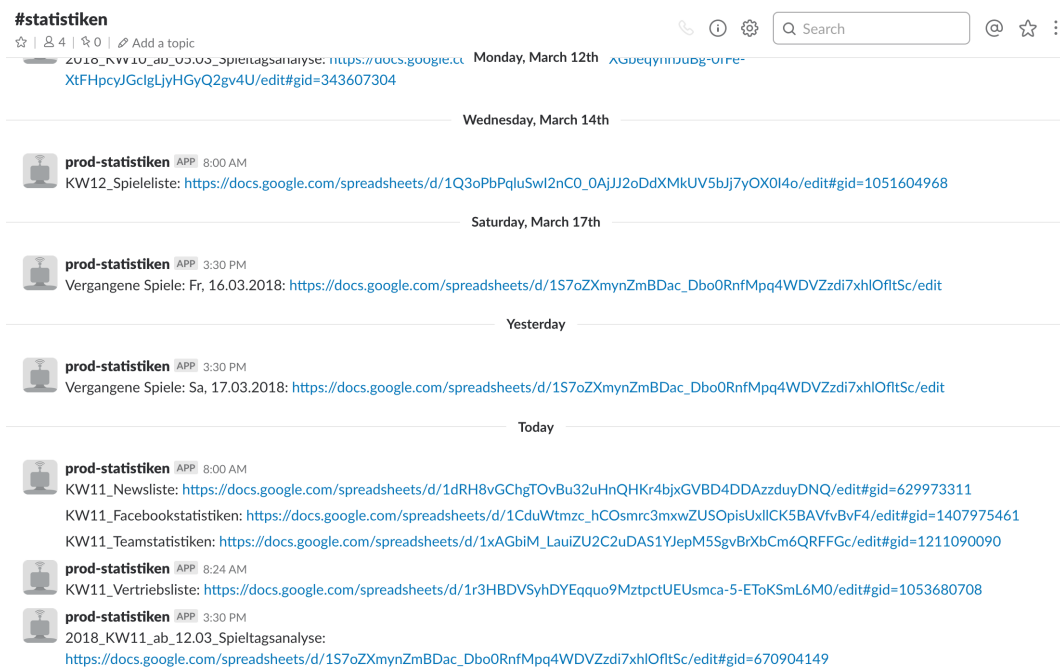


Figure 3.15.: Generated slack messages

3. Implementation

Integration in Angular application

One way to publish time series of followers and rankings are highcharts¹². Highcharts are interactive JavaScript charts for web applications.

Angular highchart¹³ is especially developed for Angular web applications and is used in this project, as can be seen in Listing 3.18.

In this implementation, time series of the followers of a specific team are requested. After the response from the API is received, the data attribute of the Angular highchart is filled with this data.

Also, the range selector is adjusted to fulfil the requirements of this project.

```
1 this.apiService.getFollowers(this.collection, this.id,
2   start_timestamp, end_timestamp).subscribe(ret => {
3
4   const followers = [];
5   for (let i = 0; i < ret['data']['followers'].length; i++)
6     {
7     const entry = [];
8     entry.push(ret['data']['followers'][i].time, ret['data']
9       ['followers'][i].followers);
10    followers.push(entry);
11  }
12  this.stock = new StockChart({
13    rangeSelector: {
14      buttons: [{
15        type: 'day',
16        count: 1,
17        text: '1T'
18      }, {
19        type: 'week',
20        count: 1,
21        text: '1W'
22      }, {
23        type: 'month',
24        count: 1,
25        text: '1M'
26      }, {
27        type: 'ytd',
```

¹²<https://www.highcharts.com>, visited on 04/01/2018

¹³<https://www.npmjs.com/package/angular-highcharts>, visited on 04/01/2018

3. Implementation

```
26         text: 'Jahr'
27     }, {
28         type: 'year',
29         count: 1,
30         text: '1J'
31     }, {
32         type: 'all',
33         text: 'Alles'
34     }],
35     inputEnabled: false,
36     selected: 5
37 },
38 title: {
39     text: 'Followers vom SV Lafnitz'
40 },
41 series: [{
42     name: 'Followers',
43     data: followers
44 }]
45 });
46 });
```

Listing 3.18: Angular highchart

4. Results

In this chapter, the outcome of this thesis will be presented, after the implementation was explained in the previous chapter.

This section shows the results of all questions that were mentioned in Chapter 1.

4.1. Reports

In order to answer questions about the activities of moderators of football teams in the frontend and activities of app users in the Android- and iOS app, reports must be generated.

The following reports are generated:

- News list report: Contains all news articles posted in each calendar week.
- Future games report: Contains all games in each calendar week until next week.
- Game statistics report: Contains general game statistics of each team.
- Matchday analysis: Contains game specific details and activities of moderators.
- Facebook report: Contains all Facebook posts of a team in relation to this project in each calendar week and in the current season.
- Month rankings report: Contains the best app users of each month.
- Rankings report: Contains the best app users of the season.
- Team statistics report: Contains general team statistics in each calendar week.

4. Results

- Sales report: Contains information of the relation between each team and this project and contact information of each team.
- Emergency list report: Contains live information about game specific details and activities of moderators.

All these mentioned Google Spreadsheets were uploaded to one of the folders in Figure 4.1.

In this section, an overview about these mentioned reports is given.

News list report

This report is updated each day and contains a list of all news articles that were written in each week. The whole report consists of one tab per week. All these tabs are saved in one Google Spreadsheet.

Such a report can be seen in Figure 4.2 and consists of the following information:

- number: Consecutive number to see how many teams have shared a news article and how many teams are public accessible.
- team: Contains the team name.
- league: The league name of the team.
- state: The state name, if the league takes place in one state.
- country: The country name of the league.
- count of news articles shared in the current season
- count of news articles shared in the current calendar week
- dates of shared news articles: Contains all dates of shared news articles in each calendar week.
- titles of shared news articles: Contains all titles of shared news articles in each calendar week.
- authors of shared news articles: Contains all authors of news articles in each calendar week.
- hits of Android app users: How many users have viewed each news article in the Android app?
- hits of iOS app users: How many users have viewed each news article in the iOS app?

4. Results













Name ↑	Zuletzt geän...
 Facebookstatistiken	19.02.2018
 Ligen- und Teamranking	09.02.2018
 Monatsranking	22.01.2018
 Newsliste	30.01.2018
 Notfalldienstliste	07.03.2018
 Spieleliste	09.02.2018
 Spielstatistiken	09.02.2018
 Spieltagsanalyse	02.11.2017
 Teamstatistiken	10.01.2018
 Vertriebsliste	19.02.2018
 Ausschicken der Reports 	21.02.2018

Figure 4.1.: Google Drive folders

4. Results

Nr.	Team	Liga	Bundesland	Land	Zeit	Titel	Verfasser	Zugriffe Android
1	FAC Wien	Erste Liga, ÖFB-Cup	Regional	Osterreich	So. 04.03.2018 20:50	Vorbericht zum Match gegen Wattens	ModeratorFAC Wien	14
2	USV Kainbach-Hönigstall II	1. Klasse Mitte A	Steiermark	Osterreich	So. 04.03.2018 20:50	2:0 Sieg in Pachern	ModeratorSkamander	12
3	SC Fürstenfeld	Landesliga, Steirer-Cup	Steiermark	Osterreich	Sa. 03.03.2018 14:15 Fr. 02.03.2018 22:00 Fr. 02.03.2018 16:30 Do. 01.03.2018 08:15	Testspielupdate Testspielupdate Spielvorschau Testspielupdate	Moderator FSK Moderator FSK Moderator FSK Moderator FSK	17 27 19 23
4	SK Vorwärts Steyr	Regionalliga Mitte, ÖFB-Cup	Regional	Osterreich	Sa. 03.03.2018 11:46 Fr. 02.03.2018 22:13 Fr. 02.03.2018 15:30 Mi. 28.02.2018 07:15 Mo. 26.02.2018 14:56	2:0 gegen Edelweiß Linz Vorwärts schlägt Austria U18 Freundschaftsspiel in Wien Testspiel gegen Austria U18 abgesagt Positive Trainingslager-Bilanz	Moderator SKVS Moderator SKVS Moderator SKVS Moderator SKVS Moderator SKVS	112 132 140 37 5
5	FC Zell am See	Salzburgerliga, Salzburger Landescup	Salzburg	Osterreich	Fr. 02.03.2018 21:09	Testspiel TSU Bramberg gegen FC Zell am See 0:0	Moderator FCZ	36
6	USV Kainbach-Hönigstall	Untertliga Mitte, Steirer-Cup	Steiermark	Osterreich	Fr. 02.03.2018 16:20	Nun sind auch wir bei Skamander!	ModeratorSkamander	7
7	SV Straß	Oberliga Mitte West, Steirer-Cup	Steiermark	Osterreich	Fr. 02.03.2018 13:53	Thomas Preschern zum SVS	Manuel Trummer	11
8	SC Leogang	2. Landesliga Süd, Salzburger Landescup	Salzburg	Osterreich	Fr. 02.03.2018 11:36	Vorbereitung SC Leogang	Moderator SC Leogang	32
9	USV	Regionalliga Mitte	Regional	Osterreich	Fr. 02.03.2018 16:50	Ergebnis nach Kick-off abgeschlossen!	Moderator SKVA	17

Figure 4.2.: News report (Google Spreadsheet)

- overall hits: How many users have clicked on each news article with the Android and iOS app?

Future games report

This report is updated every 10 minutes and contains a list of all games that take place in the last, current and next week. The whole report consists of one tab per week. All these tabs are saved in one Google Spreadsheet.

This report can be seen in Figure 4.3 and is filled with the following data:

- kick-off of the game: Contains the start date of a match.
- country: The country name of the league.
- state: The state name, if the league takes place in one state.
- league: When there exists a more precise information about the sub-league in the backend (e.g. when the sub-league is "Regionalliga Mitte", the league would be "Regionalliga").
- sub-league: The league name of the game.

4. Results

1	Zeit	Land	Bundesland	Hauptliga	Sub Liga	Ligenlink	Heim	Auswärts	Besuche	Verlängerung möglich	Spiel abgesagt
2	10.04.2018 18:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	ATSV Stadl-Paura	SC Weiz	-	nein	
3	10.04.2018 18:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	SC Kaisdorf	FC Gleisdorf 09	-	nein	
4	10.04.2018 18:30	Österreich	Regional		Erste Liga	https://www.fussballoesterreich.at/foes/Datenservice/BFV-Bundesliga-SkyGoErsteLiga/Tabelle	FAC Wien	WSG Wattens	19	nein	
5	10.04.2018 19:00	Österreich	Steiermark		Landesliga	https://www.fussballoesterreich.at/foes/Datenservice/STFV-Landesliga-LL/Tabelle	USV Gnas	SC Liezen	-	nein	
6	10.04.2018 19:00	Österreich	Steiermark		Landesliga	https://www.fussballoesterreich.at/foes/Datenservice/STFV-Landesliga-LL/Tabelle	TUS Heiligenkreuz am Waasen	FC Bad Radkersburg	10	nein	
7	10.04.2018 19:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	TUS Bad Gleichenberg	WAC Amateure	1	nein	
8	10.04.2018 19:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	SV Lafnitz	Union St. Florian	-	nein	
9	10.04.2018 19:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	DSC Deutschlandsberg	UVB Vöcklabruck	-	nein	
10	10.04.2018 19:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	UNION Gurten	USV Altheiligen	-	nein	
11	10.04.2018 19:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	LASK Juniors	SK Vorwärts Steyr	14	nein	
12	10.04.2018 19:00	Österreich	Regional	Regionalliga	Regionalliga Mitte	https://www.fussballoesterreich.at/foes/Datenservice/SFV-Regionalligen-RegionalligaMitte/Tabelle	SK Sturm Graz Amateure	SK Austria Klagenfurt	-	nein	
13	10.04.2018 19:15	Österreich	Salzburg		Salzburger Landescup		SV Straßwalchen	UFC Altenmarkt	-	ja	

Figure 4.3.: Future games report (Google Spreadsheet)

- table URL: Contains the table URL of the Austrian football association. On this page, official data about a league can be found and checked if it is similar to the data of this project.
- home team: Contains the home team of this match.
- away team: Contains the away team of this match.
- visits: How many users have clicked on this game in the Android and iOS app?
- overtime information: Is overtime possible in a game?
- cancellation information: Contains information about the cancellation of a football game.
- teams without games: Contains team names which do not have a game in a specific week.

Game statistics report

This report consists of game statistics and with this information, interesting coherences can be found.

The following statistics are calculated for the current season:

4. Results

- team: Contains the team name.
- league: The league name of the team.
- state: The state name, if the league takes place in a state.
- country: The country name of the league.
- victory (most used formation): Contains information about the most used formation, when the team wins a game (e.g. 5x: 4-4-2).
- draw (most used formation): Contains the most used formation, when the result of a game is a draw.
- defeat (most used formation): Contains the most used formation, when the team loses a game.
- home goals: Contains the sum of all home goals.
- away goals: Contains the sum of all away goals.
- ratio between home and away goals: Compares the sum of all home with the sum of all away goals. When there is a huge gap between home and away goals, this will be mentioned.
- goals in the first half: Consists of the sum of all goals during the first half.
- goals in the second half: Consists of the sum of all goals during the second half.
- ratio between first half and second half goals: Compares the sum of all goals during the first half with the sum of all goals during the second half. When there is a huge gap between them, this will be mentioned.
- sum of goals: Finally, the sum of all achieved goals in this season is given.

Matchday analysis

There is one spreadsheet in every week and it consists of a list of all games that were in the current week.

Such a report can be seen in Figure 4.4 and contains this information:

- number: All games are numbered consecutively.
- start of the match: Contains the kick-off time of the game.
- team: Contains the two teams of the match.
- country: The country name of the league of the game.
- state: The state name, if the league takes place in one state.

4. Results

Nr.	Datum	Team	Land	Bundesland	Liga	Moderation	Spielort	Endstand	Nur Endstand eingetragen	Stat Event
1	Mo. 26.02.2018 17:30	FC Deutschkreutz	International	Regional	Freundschaftsspiel	Vereinsmoderation	Heim	1		Spiel Event an
3		SV Lackenbach				Vereinsmoderation	Auswärts	0		Spiel Event an
4	Mi. 28.02.2018 18:00	SK Sturm Graz	Österreich	Regional	ÖFB-Cup	Standardmoderation	Heim	3		Spiel Event an
5		SV Wimpassing				Vereinsmoderation	Auswärts	0		Spiel Event an
6	Sa. 03.03.2018 11:00	USV Kainbach-Hönigstal	International	Regional	Freundschaftsspiel	Standardmoderation	Heim	0	ja	abw
7		SVU Gleinstätten				Vereinsmoderation	Auswärts	5		anw
8	Sa. 03.03.2018 12:00	SC Kalsdorf KM II	International	Regional	Freundschaftsspiel	Vereinsmoderation	Heim	-		anw
9		SU Rebenland				Vereinsmoderation	Auswärts	-		anw

Figure 4.4.: Matchday analysis (Google Spreadsheet)

- league: The league name of the game.
- moderation type: There is a differentiation between team moderation, fan moderation, redactional moderation and standard moderation.
- sports venue: Which team plays at home and which team plays away?
- end score: Contains the end score of the game (if the game has an end score).
- only end score entered: When only the end score and no activities were entered.
- delayed evaluation at event start: Contains information about the delayed evaluation flag. When it is true, the evaluable questions are answered not during but after the game. The event start in the Android and iOS app takes place several hours before the match starts, at this time moderators have the possibility to ask questions.
- delayed evaluation at game start: Information about the delayed evaluation flag at this time.
- moderator activity during the game: There is a differentiation between standard moderated, absent, weak activity, mediocre activity and strong activity.
- kick-off: Which moderator started at which time the football game?

4. Results

- end first half: Which moderator ended when the first half of the football game?
- start second half: Which moderator started at which time the second half of the football game?
- end second half: Which moderator ended when the second half of the football game?
- date of the game creation: When was the game created in the frontend?
- moderator of the game creation: Which moderator created the game in the frontend?
- game edited: Which moderator edited when the game?
- match sponsor changed: Was the match sponsor changed? What is the new match sponsor?
- line-up entered: Was the line-up entered? If yes, which moderator entered the line-up at which time?
- automated questions: How many automated questions were posted?
- manual entered questions: How many questions were manually added to the game?
- questions evaluated: How many questions were evaluated at which time and which moderator evaluated them?
- goal: Which moderator entered at which time goals?
- own goal: Which moderator entered at which time own goals?
- yellow card: Which moderator entered at which time yellow cards?
- red card: Which moderator entered at which time red cards?
- yellow-red card: Which moderator entered at which time yellow-red cards?
- injury: Which moderator entered at which time injuries?
- substitution: Which moderator entered at which time substitutions?
- top-chance: Which moderator entered at which time top-chances?
- top-parade: Which moderator entered at which time top-parades?
- top-tackle: Which moderator entered at which time top-tackles?
- offside goal: Which moderator entered at which time offsides?
- penalty goal: Which moderator entered at which time penalty goals?
- penalty miss: Which moderator entered at which time penalty misses?
- automated posts: How many automated posts were added automatically?
- manual entered posts before and after the game: How many posts were written at that time?

4. Results

- manual entered posts during the game: How many posts were written at that time?
- moderators: Consists of a collection of all moderators who have done anything before, during and after the game
- count of news articles in the last seven days: How many news articles were written during the last week?
- date of the last news article: At which time was the last news article written?
- last news title: What was the last title of a news article?

Facebook report

This report is updated each day and consists of an overview about all Facebook posts which a team has posted in association with this project. The whole report consists of one tab per week. All these tabs are saved in one Google Spreadsheet.

There is also another Facebook report which consists of information about all posts of a team per season and all teams that have not posted anything about this project.

Such a report can be seen in Figure 4.5 and consists of the following information:

- country: The country name of the team.
- state: The state name, if the league takes place in one state.
- league: The league name of the team.
- Facebook hyperlink of the team site: Contains a hyperlink to the Facebook team site. This site will be examined.
- all shared postings on Facebook: Here all posts that are written in association with this project are collected.
- all dates of the Facebook sharings: Here dates of all posts that are written in association with this project are collected.
- Facebook message: When a shared message on Facebook contains the keyword of this project.

4. Results

Nr.	Team	Land	Bundesland	Liga	Facebook	Facebook-Likes	Anzahl Posts mit Skamander-Bezug	Alle Facebook Sharings
1	ASK Voitsberg	Österreich	Steiermark	Landesliga	https://www.facebook.com/ASK_Voitsberg	4275	2	https://www.facebook.com/ASK_Voitsberg/posts/101562975578741 https://www.facebook.com/ASK_Voitsberg/posts/101562975186741
2	ASKÖ Klagenbach	Österreich	Burgenland	Burgenlandliga	https://www.facebook.com/ASKO-Klagenbach-238514772894185/	515	4	https://www.facebook.com/askoeklingenbach/posts/1712230012189 https://www.facebook.com/askoeklingenbach/posts/1712167458862 https://www.facebook.com/askoeklingenbach/posts/1712134032198 https://www.facebook.com/askoeklingenbach/posts/1711624592249
3	FC Eurotours Kitzbühel	Österreich	Regional	Regionalliga West	https://www.facebook.com/fckitz.at	1633	1	https://www.facebook.com/fckitz.at/posts/184280054776020
4	FC Gleisdorf 09	Österreich	Regional	Regionalliga Mitte	www.facebook.com/Fc-Gleisdorf-09-1566655350293982	2347	8	https://www.facebook.com/permalink.php?story_fbid=2071910606426118&id=151 https://www.facebook.com/permalink.php?story_fbid=2071704529780059&id=151 https://www.facebook.com/permalink.php?story_fbid=2071218906469318&id=151 https://www.facebook.com/permalink.php?story_fbid=20694562333823&id=151 https://www.facebook.com/permalink.php?story_fbid=2069443966672782&id=151 https://www.facebook.com/permalink.php?story_fbid=2069416886675510&id=151 https://www.facebook.com/permalink.php?story_fbid=2069400156677163&id=151 https://www.facebook.com/permalink.php?story_fbid=2069378983345947&id=151
5	SC Loipersdorf-Kitzsdorf	Österreich	Burgenland	1. Klasse Süd	http://facebook.com/scck.at	709	1	https://www.facebook.com/scck.at/posts/1861534237210411
6	SK Vorwärts Steyr	Österreich	Regional	Regionalliga Mitte	www.facebook.com/vorwaerts.steyr	3925	1	https://www.facebook.com/vorwaerts.steyr/posts/16933958140871
7	SK Werndorf	Österreich	Steiermark	Untere Liga	https://www.facebook.com/FliesenGarberWerndorf/	596	2	https://www.facebook.com/FliesenGarberWerndorf/posts/16878521646 https://www.facebook.com/FliesenGarberWerndorf/posts/16871868486
8	SV Flavia Solva	Österreich	Steiermark	Gebietsliga West	www.facebook.com/svflaviasolva	1618	1	https://www.facebook.com/svflaviasolva/posts/97891257894988/
9								https://www.facebook.com/evgralla/posts/2117010794983462 https://www.facebook.com/evgralla/posts/2117010794983462

Figure 4.5.: Facebook report (Google Spreadsheet)

- Facebook link: This classification contains hyperlinks about games, rankings, news, line-up, question, post and website. When the shared link cannot be classified it one of these mentioned things it can be found in miscellaneous.
- Facebook attachment: When an attachment about this project was shared.
- all combinations: Here all combinations of message, link and attachments are calculated (e.g. message and link were shared on Facebook).

A description of how this data is gathered is mentioned in Subsection 3.3.3.

Month rankings report

This report is sent out on the first day of each month and contains three tabs. In the first tab a league ranking, in the second tab a state ranking and in the third tab a country ranking can be found.

In the league ranking, the 10 best app users of each moderated league can be found. There is also a state and country ranking with the 10 best app

4. Results

users of each moderated state and country. The 10 best app users are the users with the most points.

This month ranking contains the best app users who have answered questions in the Android- and iOS app during the last month.

Rankings report

This report is sent out on the first day of each month and contains two tabs. In the first tab a league ranking and in the second tab a team ranking can be found.

In the league ranking, the 30 best app users of each moderated league can be found. There is also a team ranking with the 15 best app users of each moderated team. These are the users with the most points.

This rankings report contains the best app users of the current season. Such a ranking can be seen in Figure 4.6. In order to maintain data protection, the names and email addresses of the best app users are removed.

The month rankings and rankings report contain the following information:

- team: Contains the team name (only in the team ranking).
- league: Contains the league of the team (in the league and month ranking).
- state: The state name, if the league takes place in one state (in the league and month ranking).
- country: The country name of the league (in the league and month ranking).
- ranking position: The ranking position of the best users is given.
- first- and last name: The first- and last names of the best users are given.
- email address: The email addresses of the best users are given.
- points: The collected points in the Android and iOS-App before, during and after a football game of the best users, by answering questions, are given.

4. Results

2018_04_01_Ligen- und Teamranking

Datei Bearbeiten Ansehen Einfügen Format Daten Tools Add-ons Hilfe Letzte Änderung vor wenigen Sekunden

100% € % 0.00 123 Arial 11 B I A

	A	B	C	D	E	F	G
1	Liga	Bundesland	Land	Rang	Vor- und Nachname	Email Adresse	Punkte
2	1. Klasse Mitte A	Steiermark	Österreich	1	Max Mustermann	adresse@gmail.com	61
3	1. Klasse Mitte A	Steiermark	Österreich	2	Max Mustermann	adresse@gmail.com	54
4	1. Klasse Mitte A	Steiermark	Österreich	2	Max Mustermann	adresse@gmail.com	54
5	1. Klasse Mitte A	Steiermark	Österreich	4	Max Mustermann	adresse@gmail.com	52
6	1. Klasse Mitte A	Steiermark	Österreich	5	Max Mustermann	adresse@gmail.com	50
7	1. Klasse Mitte A	Steiermark	Österreich	6	Max Mustermann	adresse@gmail.com	48
8	1. Klasse Mitte A	Steiermark	Österreich	7	Max Mustermann	adresse@gmail.com	46
9	1. Klasse Mitte A	Steiermark	Österreich	8	Max Mustermann	adresse@gmail.com	31
10	1. Klasse Mitte A	Steiermark	Österreich	9	Max Mustermann	adresse@gmail.com	27
11	1. Klasse Mitte A	Steiermark	Österreich	10	Max Mustermann	adresse@gmail.com	17
12	1. Klasse Mitte A	Steiermark	Österreich	11	Max Mustermann	adresse@gmail.com	11
13	1. Klasse Mitte A	Steiermark	Österreich	12	Max Mustermann	adresse@gmail.com	6
14	1. Klasse Mitte A	Steiermark	Österreich	13	Max Mustermann	adresse@gmail.com	4
15	1. Klasse Mitte A	Steiermark	Österreich	14	Max Mustermann	adresse@gmail.com	2
16	1. Klasse Mitte A	Steiermark	Österreich	14	Max Mustermann	adresse@gmail.com	2
17	1. Klasse Mitte A	Steiermark	Österreich	14	Max Mustermann	adresse@gmail.com	2
18	1. Klasse Mitte A	Steiermark	Österreich	14	Max Mustermann	adresse@gmail.com	2
19	1. Klasse Ost A	Steiermark	Österreich	1	Max Mustermann	adresse@gmail.com	209
20	1. Klasse Ost A	Steiermark	Österreich	2	Max Mustermann	adresse@gmail.com	90
21	1. Klasse Ost A	Steiermark	Österreich	3	Max Mustermann	adresse@gmail.com	62
22	1. Klasse Ost A	Steiermark	Österreich	4	Max Mustermann	adresse@gmail.com	50
23	1. Klasse Ost A	Steiermark	Österreich	5	Max Mustermann	adresse@gmail.com	48
24	1. Klasse Ost A	Steiermark	Österreich	6	Max Mustermann	adresse@gmail.com	42
25	1. Klasse Ost A	Steiermark	Österreich	7	Max Mustermann	adresse@gmail.com	41
26	1. Klasse Ost A	Steiermark	Österreich	8	Max Mustermann	adresse@gmail.com	38

Ligenranking Teamranking

Figure 4.6.: Rankings report (Google Spreadsheet)

4. Results

Team statistics report

This report is updated each day and contains a list of all teams. The whole report consists of one tab per week. All these tabs are saved in one Google Spreadsheet.

- team: Contains the team name.
- league: The league name of the team.
- state: The state name, if the league takes place in one state.
- country: The country name of the league.
- status activation: Is this team public accessible in the app (test-team or not)?
- moderation: There is a differentiation between team moderation, fan moderation, redactional moderation and standard moderation.
- team logo uploaded: Is a team logo uploaded or not?
- moderators: Consists of all enabled moderators of this team.
- followers: Contains the followers count of the team in the Android- and iOS app.
- users in season ranking: Contains the count of all users who answered at least one question in the current season.
- standard moderated (home): Contains the count of all standard moderated games at home in the current season.
- absent (home): Count of all games at home in the current season where the moderator was classified as absent.
- weak moderation (home): Count of all games at home in the current season where the moderation was classified as weak moderation.
- mediocre moderation (home): Count of all games at home in the current season where the moderation was classified as mediocre moderation.
- strong moderation (home): Count of all games at home in the current season where the moderation was classified as strong moderation.
- standard moderated (away): Contains the count of all standard moderated games at away in the current season.
- absent (away): Contains the count of the moderator activity classification for the away team.
- weak moderation (away)
- mediocre moderation (away)

4. Results

- strong moderation (away)
- standard moderated (overall)
- absent (overall): Contains the count of the moderator activity classification for the home and away team.
- weak moderation (overall)
- mediocre moderation (overall)
- strong moderation (overall)
- count of news articles in this season
- date of the last news article: With this information it can be said how active a moderator writes news entries.
- count of players in the squad: With this information it can be said that the data is up-to-date or not.
- date of the last change of one player in the squad
- trainer name: Contains the trainer name of this team, when it was entered in the frontend.
- Twitter: Contains the Twitter page of this team, when it was entered in the frontend.
- Instagram: Contains the Instagram page of this team, when it was entered in the frontend.
- YouTube: Contains the YouTube account of this team, when it was entered in the frontend.
- main sponsor: Contains the count of the main sponsors and the main sponsor names, when a main sponsor was entered in the frontend.
- premium sponsor: Contains the count of the premium sponsors and the premium sponsor names.
- standard sponsors: Contains the count of the standard sponsors and the standard sponsor names.

Sales report

This report is sent out once a week. The whole report consists of one tab per week. All these tabs are saved in one Google Spreadsheet.

It contains the following information of each team:

- country: The country name of the team.
- state: The state name, if the league takes place in one state.

4. Results

- league: The league name of the team.
- sub-league: The sub-league of the team, when there is a sub-league (e.g. the sub-leagues of „Regionalliga“ are „Regionalliga Mitte“, „Regionalliga Ost“ and „Regionalliga West“).
- test-team: Is this team public accessible in the app or at the moment a test-team?
- start date publication: When was this team enabled for public use in the app?
- end date publication: When was this team disabled for public use in the app?
- moderated: Has this team a moderator or not?
- start date moderation: At which time started the moderation of a team?
- end date moderation: At which time ended the moderation of a team?
- name, phone number and email of the chairman of the club, of the person responsible for this project in the club and of the moderators of the club

Emergency list report

This emergency report is updated every 10 minutes in a Google Spreadsheet. It is used on a matchday to observe the current status of each game.

The whole report consists of one tab per day in a specific week, when football matches take place. All these tabs are saved in one Google Spreadsheet.

Such a report can be seen in Figure 4.7 and consists of the following information:

- number: All games are numbered consecutively.
- kick-off: Contains the kick-off time of the game.
- team: Contains the two teams that clash each other.
- country: The country name of the league of the game.
- state: The state name, if the league takes place in one state.
- league: The league name of the game.
- moderation: There is a differentiation between team moderation, fan moderation, redactional moderation and standard moderation.

4. Results

Nr.	Datum	Team	Moderation	Aktueller Spielstand	Halbzeitstand	Endstand	Ligenseite	Nur Endstand eingetragen	Anpfiff durch	Ende HZ 1
1	Mi. 04.04.2018 19:00	SC Kalsdorf KM II	Vereinsmoderation	beendet	nicht getickert	3 (nachgetragen: ModeratorSkamander um 20:59)	https://www.fussballoesterreich.at/foes/Datenservice/STEV-SparkassenUnterliga-ULM/Tabelle			
3		SV Hausmannstätten	Standardmoderation	beendet	nicht getickert	0 (nachgetragen: ModeratorSkamander um 20:59)	https://www.fussballoesterreich.at/foes/Datenservice/STEV-SparkassenUnterliga-ULM/Tabelle			
4	Mi. 04.04.2018 19:30	USV Kainbach-Höfnigal	Vereinsmoderation	beendet	3	4	https://www.fussballoesterreich.at/foes/Datenservice/STEV-SparkassenUnterliga-ULM/Tabelle		UKH Fan #1 (19:33)	UKH Far (20:21)
5		SC Unterpriessstätten	Standardmoderation	beendet	1	1	https://www.fussballoesterreich.at/foes/Datenservice/STEV-SparkassenUnterliga-ULM/Tabelle			

Figure 4.7.: Emergency list report (Google Spreadsheet)

- sports venue: Which team plays at home and which team plays away?
- current minute: Contains the current minute of a game (e.g. 15 - 30 (18))
- current score: Contains the end score of the game (if the game has an end score).
- first half end score: Contains the end score of the first half.
- second half end score: Contains the end score of the second half.
- end score after the first half of the overtime
- end score after the second half of the overtime
- end score after the penalty shootout
- league URL: Contains the table URL of the league, where the match takes place, from the Austrian football association.
- only end score entered: When only the end score and no activities were entered.
- kick-off information: Which moderator started when the football game?
- end first half: Which moderator ended when the first half of the football game?
- start second half: Which moderator started when the second half of

4. Results

the football game?

- end second half: Which moderator ended when the second half of the football game?
- start first half of the overtime: Contains moderator information and the timestamp of this action.
- end first half of the overtime: Contains moderator information and the timestamp of this action.
- start second half of the overtime: Contains moderator information and the timestamp of this action.
- end second half of the overtime: Contains moderator information and the timestamp of this action.
- start penalty shootout: Contains moderator information and the timestamp of this action.
- end penalty shootout: Contains moderator information and the timestamp of this action.
- line-up entered: Was the line-up entered? If yes, which moderator entered the line-up at which time?
- questions evaluated: How many questions were evaluated at which time and which moderator evaluated them?
- goal: Which moderator entered when the goal activities and which text was entered?
- own goal: Which moderator entered when the own goal activities and which text was entered?
- all goals entered: Checks if the final score corresponds with the entered goal and own goal activities.
- yellow card: Which moderator entered when the yellow card activities and which text was entered?
- red card: Which moderator entered when the red card activities and which text was entered?
- yellow-red card: Which moderator entered when the yellow-red card activities and which text was entered?
- injury: Which moderator entered when the injury activities and which text was entered?
- substitution: Which moderator entered when the substitution activities and which text was entered?
- top-chance: Which moderator entered when the top-chance activities and which text was entered?

4. Results

- top-parade: Which moderator entered when the top-parade activities and which text was entered?
- top-tackle: Which moderator entered when the top-tackle activities and which text was entered?
- offside goal: Which moderator entered when the offside activities and which text was entered?
- penalty goal: Which moderator entered when the penalty goal activities and which text was entered?
- penalty miss: Which moderator entered when the penalty miss activities and which text was entered?
- end score two hours after game start correct: Contains a query to check the end score very fast.
- all goals entered two hours after game start: Contains a query to check if all goals were entered.
- questions evaluated two hours after game start: Contains a query to check if all questions were evaluated.
- users in ranking: Contains the count of all playing users who has answered at least one question in the Android- and iOS app.
- viewers: Contains the count of all viewers of the game in the app.
- match sponsor changed: When the match sponsor is changed, the sponsor name is displayed.
- Facebook site: Contains the Facebook site of the team, when there is one entered in the frontend.
- Facebook sharings between event start and event end: Contains hyperlinks of all Facebook sharings on the matchday to check if the team uses this project's sharing possibilities.
- date of these Facebook sharings
- count of these Facebook sharings

4.2. Frontend

There is already an existing administration web application, where users, teams, leagues and templates can be managed. This administration web application was extended with the logged information mentioned in Section 3.2.

4. Results

In this chapter, the integrated data in the frontend will be described.

News views

After the logging of news articles is described in 3.2.2 and how this data, which should be displayed, is explained in 3.3.1, this information can be viewed in the Angular application.

A news article is displayed with the information of the count of the users who have viewed this specific article. A news article is viewed when app users have clicked on a notification, on a deep link and when an app user clicks on a news article in the Android- and iOS app.

Sponsor clicks

The information about sponsor clicks is similar handled as the news views information. The logging is mentioned in 3.2.2 and the calculated data is described in 3.3.1.

With this data it can be said how many Android- and iOS app users have clicked on each main, premium and standard sponsor.

This data is displayed beside each sponsor name.

Time series

The logging of time series is described in 3.2.3 and the calculation of the displayed data in 3.3.1.

These time series are presented for:

- Followers: Contains the followers count.
 - Team
 - League
- Rankings: Contains the count of users (with registration) and anonymous users in ranking.

4. Results

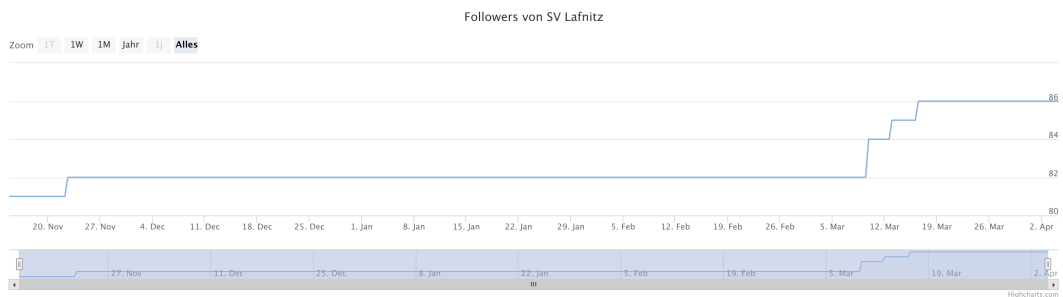


Figure 4.8.: Angular 5 application displaying time series of followers

- Game: Time series of each game ranking are displayed. Contains additionally viewers and visits of a game in the Android- and iOS app.
- Team: Time series of each team ranking are displayed.
- League: Time series of each league ranking are displayed.
- State: Time series of each state ranking are displayed.
- Country: Time series of each country ranking are displayed.
- Global: Time series of all app users is displayed.

An example for the outcome of time series of the followers of a specific team can be seen in Figure 4.8.

5. Discussion

This chapter shows the discussion of all results that were presented in Chapter 4.

5.1. Reports

Reports are generated to answer a lot of questions in relation to moderators of football teams in the frontend and app users in the Android- and iOS app of this project.

News list report

In order to answer the question how many news articles have been written by moderators, a news report is generated.

It shows the activity of a moderator. When a moderator does not use this service anymore, the support team of this project can react to such problematic cases. News articles are very important to publish matchday or injury related information. This report helps also to understand a football club better, because every author of a news article can be seen.

Additionally, the importance of each team can be evaluated, because the visits by Android- and iOS users and the overall visits are given.

5. Discussion

Future games report

In the future games report, information about all games which take place in every calendar week, until next week, can be found.

With this information it can be examined if a moderator of a team has created future games in the frontend of this project. It is possible to check the cancellation status of a game, the visits of this game in the apps and if overtime is possible.

Teams without games in a specific week can also be double-checked with a hyperlink of the Austrian football association, which contains official data.

Game statistics report

With the data of this report, interesting statistics about games can be delivered, e.g. home and away goals or first half and second half goals can be compared. When there are huge gaps between these goals, this information can be highlighted.

These statistics can be integrated in packages for football teams they can get when they are willing to pay something.

Matchday analysis

This report consists of information about matchdays and is sent out after each matchday.

With this information, the support team of this project can check if the moderators of the teams have done all things they must do before, during and after a game in the frontend like evaluating questions, providing goal scorers, entering the line-up and a lot of other things fast and easily.

Also, an evaluation of the activities of a moderator is given, there is a classification in absent, standard moderated, weak moderation, mediocre moderation and strong moderation. Moderators with many activities before, during and after a game can be honoured.

5. Discussion

Facebook report

With this data, Facebook posts of all teams which are part of this project can be examined per calendar week. There is also the possibility to check all Facebook posts of this season and if this project is mentioned.

Teams without posting anything can be detected and it can be suggested to use the sharing possibilities this project gives in order to share things on Facebook like publishing a news article.

After receiving this report, the importance of each team can be rated, because it is possible to see all people's reactions to these Facebook posts. It is also possible to see how many people liked the Facebook page of a specific team.

The most popular Facebook posts can also be examined. The most popular posts are these with the most reactions (like, love, haha, wow, sad, angry, shares) of people on Facebook. The shared hyperlinks are classified as mentioned in Section 4.1.

Month rankings report

This month ranking shows 10 app users with most points in every league, state and country of this project's Android and iOS app during the last month. These users must have answered at least one question during the last month.

With these results, app users can be honoured that were highly motivated to use the app in order to answer questions in the last month. The best users can be asked to be a fan moderator of a specific team or a whole league. There is also the possibility to redactional moderate a team.

Rankings report

This rankings report shows 30 app users with most points in league ranking and 15 app users with most points at a team ranking during the whole

5. Discussion

season. These users must have answered at least one question during the season.

At the end of the season, the best fans of a team can be acknowledged with awards and they can also be asked to redactional moderate a team or be a fan moderator.

Team statistics report

With this report, the whole activities of a moderator in the frontend can be evaluated.

This activity can be rated as follows:

- News articles: How many news articles are written during the whole season?
- Facebook sharings: How many fans has a team on Facebook? How many Facebook sharings are written about this project during the whole season and what are the reactions of fans to these posts?
- Activities on matchday: What has a moderator done during a season on each matchday? Did he provide a live ticker, interesting questions and answers to these questions, write general posts, enter the line-up and many other activities?

It is also possible to check if a team logo is uploaded, which moderators of a football club are enabled, how many followers a team has, which sponsors are entered and a lot of other information.

In order to provide up-to-date squad data, the last modification of a squad can be checked and how many players are part of each team.

Sales report

This report consists of sales-oriented information, e.g. start and end date of the moderation of a team. It also contains all contact details of a team like name, phone number and email of the representative, responsible person of a football club for this project and the enabled moderators.

5. Discussion

It helps people of this project's sales and support team in order to contact teams when there are new features provided by the frontend or e.g. when the responsible person for this project of a team did not do anything in the frontend anymore.

Emergency list report

It is used on a matchday to observe the current status of each game. With this list it is easy to detect errors very fast, because it is updated every ten minutes.

The current score, half time score, end score, live ticker text and a lot of other information can be checked about a specific game. With this information it can be examined what a moderator has done on a matchday. It is also possible to check Facebook posts of teams on a matchday.

This report helps to check whether a team moderator moderate the game of his team live. Also, the quality of a live ticker can be evaluated, and the best moderators can be honoured with this information.

5.2. Frontend

In this section, all statistics that are integrated in the existing Angular 5 web application, called administration view, are discussed.

In order to publish the logged information, statistics about news views and sponsor clicks and time series of followers and rankings are displayed.

News views

Every team can see how often a news article was visited by Android-, iOS- and overall app users.

5. Discussion

With this data, they can evaluate the importance of this project for them and can improve their performance. For example, when they see there are very high access figures, teams can write more news articles.

Sponsor clicks

This project often receives requests from teams to share the following information from the Android and iOS-app regularly:

- count of clicks on the information tab of a team: How many times was the info tab of the team visited? In this tab, all premium, main and normal sponsors are listed.
- count of sponsor clicks: How often have app users clicked on the sponsor hyperlinks in the info tab of the team?

All this data is integrated in the administration view and with the information of this thesis, all these questions can be answered, and this information can be sold.

Consequently, also the teams can sell this reports to their sponsors.

Time series

An example of the results of this thesis can be seen in Figure 4.8, where time series of the followers of a specific team are displayed.

With such information it is possible to make historical and predictive analysis of this team. In this example, the followers count is continuously rising and shows that the moderator of the team "SV Lafnitz" does a quite good job.

After taking a closer look on the work of the moderator of the team, it is clear why this occurs. He shares daily news articles and other social media like Facebook.

He also asks interesting questions before, during and after a football game and provides a live ticker with entered goals, own goals, substitutions and

5. Discussion

other activities during a game. He also updates the squad of the team regularly.

6. Conclusions

Finally, all conclusions are presented in this chapter. In order to summarize this thesis, the research questions will be evaluated, raised problems will be given and future work will be described.

6.1. Evaluation of the Research Questions

In this section, the research questions, which were raised in the beginning of this thesis, are evaluated:

- “How and with which components is it possible to save a great amount of user data and analyse it under the usage of the current State of the Art?”

The outcome of a research about useful components was to use Amazon Web Services, when they are already in use. In such cases, the process and data flow can be optimised very easily.

A very smart process was developed in order to achieve this, the data flow is presented in Figure 3.10.

The Android- and iOS apps log the in Chapter Appendix A displayed actions to track all actions all users do in the app.

The apps send these actions to the backend’s API after 50 logged entries, when the user closes the app and when the logout is used. After receiving this data at the backend, it is published to an Amazon Web Service. Amazon Simple Queue Service (SQS) takes these messages. In Apache Storm, a spout is defined which waits for messages from Amazon SQS. After receiving such messages, they are handled in the defined bolt.

There are three possible scenarios handled in the bolt:

6. Conclusions

- When receiving sponsor clicks actions, they are saved in the analytics-MongoDB with the userid in order to obtain the sum of all clicks on a specific sponsor via MongoDB.
- Also, when receiving news viewing actions, they are saved in the analytics-MongoDB to obtain how many users have seen a specific news article.
- All actions are transported to Amazon Kinesis Firehose which stores the data in Amazon S3 buckets in a structured way. There are files containing all logged actions per hour, day, month and year.

In order to query Amazon S3 buckets, also an Amazon Web Service is used. With Amazon Athena it is possible to access Amazon S3 buckets directly and execute queries on them. Amazon Athena is based on Presto, a distributed SQL query engine for big data.

- “How is it possible to save time series and display them with time series analysis?”

Time series are collected for rankings (game, team, league, state, country and global) and followers (team and league). With time series it is very easy possible to analyse this data. In order to make historical and predictive analyse, a very intuitive solution must be found.

Time series are saved in a very efficient way in the analytics-MongoDB. The game ranking will be checked every five minutes before and after the game and in every minute during the game. When users in ranking, anonymous users, viewers or visits have been changed during this time, a new entry will be logged in the MongoDB.

Teams, leagues, states, countries and global rankings are checked every hour and when it has been changed, the new entry will be logged.

Follower counts will also be checked and possibly logged every hour. After saving time series of rankings and followers the data must be displayed. During a research, a lot of possibilities to display such data were found. A constraint was that Angular 5 was current in use. Consequently, time series must be developed in Angular and not with other frontend frameworks like ReactJS.

Finally, highcharts¹ was selected. It provides very powerful ways

¹<https://www.highcharts.com>

6. Conclusions

to achieve this and is already in use by many big companies like Facebook, Nokia and Visa. An example of time series displayed with highcharts can be seen in Figure 4.8.

6.2. Limitations

The work of this thesis makes it possible to collect app actions. With more collected data, e.g. after logging the Android-, iOS- and Web app for a whole season, more analysis can be done. A big problem was that the new season proceeded not until the begin of March. Consequently, there are not many logged actions at the moment.

Another limitation was that the iOS app developers of this project have not logged all actions, which are mentioned in Chapter Appendix A, yet.

When collecting data over a lot of games and over a season, the whole process can be optimized. It is also possible that other interesting coherences will be found in future. Also, clearer statements can be made after analysing the logged information.

6.3. Future work

A lot of work has been done and a lot of work must be done in future in order to fulfil the requirements of this project for having a great future.

Here a list of several future tasks will be mentioned that could be interesting for this project:

- Clickstream analysis: A very huge topic will be clickstream analysis. The path of an app user in the Android- and iOS app is already tracked. This data could be analysed in order to optimize the app. A possible outcome could be that almost all app users do not use a specific app function. Consequently, this function must be highlighted to the user. Another interesting outcome would be that users try specific things more often than other things like viewing the current ranking

6. Conclusions

during a game.

- **Markov chain:** A corresponding scientific topic would be the Markov model. With a Markov chain it is possible to predict the next action of a user.
In such a model there exists a set of states. In the context of this project, possible states would be answering questions, clicking on the activities tab in a game in the app etc. With the Markov chain, the probabilities of the transition from one state to another state are calculated. The received data could also be graphically processed in order to analyse it better.
- **Integration of live data from football games:** This project will receive live data from the Austrian Football Association. In detail this package contains information about teams, leagues, states, countries and general player information.
With this data, moderators of football clubs must not spend that much time in the frontend, because a lot of the work they have to do at the moment will be automated. They do not have to create games in the frontend and also all match specific activities like goals, own goals and yellow cards will be loaded automatically. After loading these activities, live questions, e.g. about the future score, can also be asked automatically.
- **Moderator ranking:** With the logging information of this work, moderator rankings could be created in order to award the best moderators. When the best moderators receive prizes, they would be highly motivated to continue their good work and also other moderators would be more encouraged to do more in the frontend.
- **Integration of videos:** A very nice feature for the future would be the creation of a videos section. In that case fans would feel a closer connection to their teams. Moderators could be awarded when uploading such live videos in the frontend.
- **Generate images:** With the received information of this thesis, interesting images could be created and afterwards shared on social media (Facebook, Twitter, Instagram, YouTube) to increase the influence of

6. Conclusions

this project in public relation.

For example, general game statistics like the difference between first half and second half goals in the current season could be published.

- Providing all collected information to teams: All teams, which are part of this project, are quite likely interested in the evaluations, which are collected for this thesis. When they are prepared to spend money for receiving this information, the outcome could be really interesting for the football teams.
- Calculate game statistics: After collecting a lot of game statistics like goals, own goals, yellow cards and other activities, this data can be analysed in many ways in order to deliver statistics about home and away games. The outcome of such statistics could be unexpected and surprising.
- Detect specific things: After providing a service for teams to report statistics they would like to know this project can offer delivering these statistics against the payment of a fee.
- Live evaluation: Another nice feature would be the live evaluation of visitor numbers. In cases where many app users are currently online in the Android- and iOS app, this information can be reported to the apps. Based on current hits, premium, main and normal sponsors can be displayed.
- Optimizing the generated reports: Over time, the requirements of this project are changing, and also other information could be interesting for reports.
Also, specific reports can be optimized. For example, the matchday analysis report can be removed, because a lot of the data of this report is integrated in the emergency list report which is updated every 10 minutes.

Appendix

Appendix A.

Logging in the app

All logged actions by the Android- and iOS apps ordered by relevant categories can be found in this chapter.

Table A.1 shows all general app actions like open and close, login and logout and many others.

action	category	id	type	timestamp	sessionid
open	app				
close	app				
login	app		email		
login	app		facebook		
login	app		guest		
logout	app		email		
logout	app		facebook		
logout	app		guest		
register	app		email		
register	app		facebook		
register	app		guest		
rate	app		android		
rate	app		ios		

Table A.1.: Logged app actions

Table A.2 shows all click actions a user does in the menu of the app.

Appendix A. Logging in the app

action	category	id	type	timestamp	sessionid
click	menu		news		
click	menu		events		
click	menu		teams		
click	menu		leagues		
click	menu		settings		

Table A.2.: Logged menu actions

Table A.3 shows all click actions a user does after receiving a notification. The action “app open” is at same time possible.

action	category	id	type	timestamp	sessionid
click	notification	newsid	news		
click	notification	eventid	event		
click	notification	eventid	goal		
click	notification	eventid	end-rank		
click	notification		info		

Table A.3.: Logged notification actions

Table A.4 shows all deep link click actions of a user, e.g. when a moderator shares a news article, he has written in the frontend, on Facebook and a user clicks on this link. The “app open” activity is at same time possible.

action	category	id	type	timestamp	sessionid
click	deeplink	newsid	news		
click	deeplink	eventid	event		
click	deeplink	teamid	team		
click	deeplink	eventid	ranking-event		
click	deeplink	teamid	ranking-team		
click	deeplink	leagueid	ranking league		

Table A.4.: Logged deep link actions

Table A.5 shows all click actions of a user in the settings menu item, when a user tries to change the email, name, password or wants to give feedback.

Appendix A. Logging in the app

action	category	id	type	timestamp	sessionid
click	settings		email		
click	settings		name		
click	settings		password		
click	settings		feedback		
click	settings		support		

Table A.5.: Logged settings actions

Table A.6 shows all actions a user does with news. A news article is viewed when the view time is equal or greater than 0.5 seconds. There is a possibility to share a news article via Facebook or another method. For successfully shared items via Facebook there exists a callback.

action	category	id	type	timestamp	sessionid
view	news	newsid	newsscreen		
view	news	newsid	teamscreen		
view	news	newsid	leaguescreen		
open	news	newsid	newsscreen		
open	news	newsid	teamscreen		
open	news	newsid	leaguescreen		
click	news	newsid	share		
share	news	newsid	facebook		
share	news	newsid	other		

Table A.6.: Logged news actions

Table A.7 shows all actions a user does in relation with a football game in the frontend.

Appendix A. Logging in the app

action	category	id	type	timestamp	sessionid
view	event	eventid	eventscreen		
view	event	eventid	teamscreen		
view	event	eventid	leaguescreen		
open	event	eventid	eventscreen		
open	event	eventid	teamscreen		
open	event	eventid	leaguescreen		
click	event	eventid	fanwall		
click	event	eventid	sponsor		
click	event	eventid	overview		
click	event	eventid	ranking		
click	event	eventid	change-team		
click	event	eventid	notifications		
click	event	eventid	lineup		
click	event	eventid	lineup-details		
click	event	eventid	ranking-details		
click	event	eventid	share		
change	event	eventid	change-team		
change	event	eventid	notifications		
share	event	eventid	facebook		
share	event	eventid	other		

Table A.7.: Logged event actions

In Table A.8, all question actions can be found. When a user clicks on a question, the type “details” is logged.

action	category	id	type	timestamp	sessionid
view	question	questionid			
click	question	questionid	answer		
click	question	questionid	points		
click	question	questionid	details		
click	question	questionid	share		
share	question	questionid	facebook		
share	question	questionid	other		

Appendix A. Logging in the app

Table A.8.: Logged question actions

In Table A.9, all post actions, a user does in the app, can be found.

action	category	id	type	timestamp	sessionid
view	post	postid			
click	post	postid	share		
share	post	postid			
share	post	postid	other		

Table A.9.: Logged post actions

All activity actions can be seen in Table A.10. An activity in a game is a goal, own goal, yellow card, red card, yellow-red card, injury, substitution, top chance, top parade, top tackle, offside, penalty goal and penalty miss.

action	category	id	type	timestamp	sessionid
view	activity	activityid			
share	activity	activityid			

Table A.10.: Logged activity actions

Table A.11 shows all team actions a user can do.

Appendix A. Logging in the app

action	category	id	type	timestamp	sessionid
view	team	teamid			
open	team	teamid	event		
open	team	teamid	teams		
open	team	teamid	league		
open	team	teamid	news		
click	team	teamid	news		
click	team	teamid	events		
click	team	teamid	squad		
click	team	teamid	info		
click	team	sponsorid	sponsor		
click	team	teamid	table		
click	team	teamid	ranking		
click	team	teamid	notifications		
click	team	teamid	add		
click	team	teamid	remove		
change	team	teamid	notifications		
share	team	teamid			

Table A.11.: Logged team actions

In Table A.12, actions in relation with a sponsor are logged like clicking on a sponsor.

action	category	id	type	timestamp	sessionid
view	sponsor	sponsorid			
click	sponsor	sponsorid			
view	page		name		

Table A.12.: Logged sponsor actions

Table A.13 shows all league actions that are logged.

Appendix A. Logging in the app

action	category	id	type	timestamp	sessionid
view	league	leagueid			
open	league	leagueid	event		
open	league	leagueid	team		
open	league	leagueid	table		
click	league	leagueid	news		
click	league	leagueid	events		
click	league	leagueid	info		
click	league	leagueid	table		
click	league	leagueid	ranking		
click	league	leagueid	notifications		
click	league	leagueid	add		
click	league	leagueid	remove		
share	league	leagueid			

Table A.13.: Logged league actions

Within a game in the Android- and iOS app, ads are displayed. All ad actions can be found in Table A.14 like viewing an ad.

action	category	id	type	timestamp	sessionid
view	ad	adid			
view	ad		notfound		

Table A.14.: Logged ad actions

In the info tab, all social media channels and sponsors are listed. Also, all clicking actions in this screen are logged, as can be seen in Table A.15.

Appendix A. Logging in the app

action	category	id	type	timestamp	sessionid
click	info	teamid/leagueid	ranking		
click	info	teamid/leagueid	table		
click	info	teamid/leagueid	homepage		
click	info	teamid/leagueid	facebook		
click	info	teamid/leagueid	twitter		
click	info	teamid/leagueid	email		
click	info	teamid/leagueid	instagram		
click	info	teamid/leagueid	youtube		

Table A.15.: Logged info actions

Bibliography

- Amazon Web Services (2018[a]). *Cloud Products*. URL: https://aws.amazon.com/products/?nc2=h_q1_p&awsml=q1-1 (visited on 03/06/2018) (cit. on p. 18).
- Amazon Web Services (2018[b]). *What is Cloud Computing?* URL: https://aws.amazon.com/what-is-cloud-computing/?nc1=h_ls/ (visited on 03/01/2018) (cit. on p. 17).
- Angular (2018[a]). *Description*. URL: <https://angular.io/guide/architecture> (visited on 03/03/2018) (cit. on p. 27).
- Angular (2018[b]). *Description*. URL: <https://angular.io/resources> (visited on 03/03/2018) (cit. on p. 27).
- Apache Storm (2018[a]). *Apache Storm*. URL: <http://storm.apache.org/> (visited on 02/15/2018) (cit. on p. 21).
- Apache Storm (2018[b]). *Concepts*. URL: <http://storm.apache.org/releases/current/Concepts.html> (visited on 02/15/2018) (cit. on p. 21).
- Box, George P. et al. (2015). *Time series analysis: Forecasting and Control*. Fifth Edition. John Wiley and Sons (cit. on p. 41).
- Braude, Eric J. and Michael E. Bernstein (2016). *Software Engineering: Modern Approaches, Second Edition*. second edition. Waveland Press (cit. on pp. 14, 15).
- C Sharp Corner (2018). *Top 5 Trending Front-End Frameworks In 2018*. URL: <https://www.c-sharpcorner.com/article/top-5-trending-front-end-frameworks-in-2018/> (visited on 04/02/2018) (cit. on pp. 26, 27).
- Chaniotis, Ioannis K., Kyriakos-Ioannis D. Kyriakou, and Nikolaos D. Tselikas (2014). *Is Node.js a viable option for building modern web applications? A performance evaluation study*. Springer (cit. on p. 24).
- Chari, Kaushal and Manish Agrawal (2017). *Impact of incorrect and new requirements on waterfall software project outcomes*. Springer Science+Business Media New York 2017 (cit. on p. 14).

Bibliography

- Choetkiertikul, Morakot et al. (2017). *Predicting Delivery Capability in Iterative Software Development*. IEEE (cit. on p. 6).
- Evans, Shawna (2017). *Angular JS For Busies*. CreateSpace Independent Publishing Platform (cit. on p. 27).
- Google (2018). *REST Resource: spreadsheets*. URL: <https://developers.google.com/sheets/api/reference/rest/v4/spreadsheets> (visited on 03/20/2018) (cit. on p. 61).
- Iqbal, Muhammad Hussain and Tariq Rahim Soomro (2015). *Big Data Analysis: Apache Storm Perspective*. International Journal of Computer Trends and Technology (IJCTT) (cit. on p. 20).
- Leavitt, Neal (2010). *Will NoSQL Databases Live Up to Their Promise?* IEEE (cit. on p. 24).
- MongoDB (2018[a]). *MongoDB for Time Series Data*. URL: <https://www.mongodb.com/presentations/mongodb-time-series-data> (visited on 03/19/2018) (cit. on p. 41).
- MongoDB (2018[b]). *NoSQL Databases Explained*. URL: <https://www.mongodb.com/nosql-explained> (visited on 03/13/2018) (cit. on pp. 24, 25).
- Node.js (2018). *Description*. URL: <https://nodejs.org/en/about/> (visited on 02/15/2018) (cit. on p. 23).
- Rosero, Raul H., Omar S. Gomez, and Glen Rodriguez (2017). *An approach for regression testing of database applications in incremental development settings*. IEEE (cit. on p. 17).
- Tilkov, Stefan and Steve Vinoski (2010). *Node.js: Using JavaScript to Build High-Performance Network Programs*. IEEE (cit. on p. 23).
- Veen, Jan Sipke van der et al. (2015). *Dynamically Scaling Apache Storm for the Analysis of Streaming Data*. IEEE (cit. on p. 21).