

Umfeld, Ausgangslage

Das [RiverSurfJam](#) in Thun bietet begeisterten Flusssurfern und Flusssurferinnen die Möglichkeit, gegeneinander anzutreten und auf der Aare ihr Können unter Beweis zu stellen.

River Surfen ist eine relativ junge Sportart in der Schweiz, mit noch geringer Popularität. Dementsprechend ist die Organisation, die Planung und die Buchführung des Events sehr simpel gehalten.

Die Zuteilung in Heats und Runden, sowie die Erfassung der Resultate werden noch manuell durchgeführt.

2020

Round 1 Men Division

| Heat 1 | | Heat 2 | | Heat 3 | | Heat 4 | | Heat 5 | | Heat 6 | |
|---------------|---|---------------|---|---------------|---|---------------|---|----------------|---|---------------|---|
| Riders | | Riders | | Riders | | Riders | | Riders | | Riders | |
| Dimitri S. | 1 | Severin K. | 1 | Sandro S. | 2 | Nicolas W. | 2 | Vince S. | 1 | Lucas B. | 1 |
| David C. | 2 | Simon B. | 2 | Parsa F. | 3 | Muli B. | 3 | Durbi C. | 2 | Dave H. | 2 |
| Dave S. | 3 | Luca T. | 3 | Luca A. | 1 | Ruby | 1 | Thomas K. | 3 | Luca H. | 3 |
| Eusebio R. | 4 | | | Vinzenz N. | | Yves A. | 4 | Christopher F. | | | |
| Judges | | Judges | | Judges | | Judges | | Judges | | Judges | |
| Lukas Ö. | | Dimitri S. | | Severin K. | | Sandro S. | | Nicolas W. | | Vince S. | |
| Dave H. | | David C. | | Simon B. | | Muli B. | | Durbi C. | | Dave H. | |
| Luca H. | | Dave S. | | Luca T. | | Ruby | | Thomas K. | | Luca H. | |
| Eusebio R. | | | | Vinzenz N. | | Yves A. | | Christopher F. | | | |

Mit steigender Beliebtheit der heutigen Randsportart besteht das Bedürfnis, eine Applikation zu entwickeln, welche die Organisation sowie die Buchführung erleichtert und die Zuschauer mit den Athleten (Riders genannt) verbindet.

Weiter zu: Aufgabenstellung & Ziel des Projektes

Aufgabenstellung & Ziel des Projektes

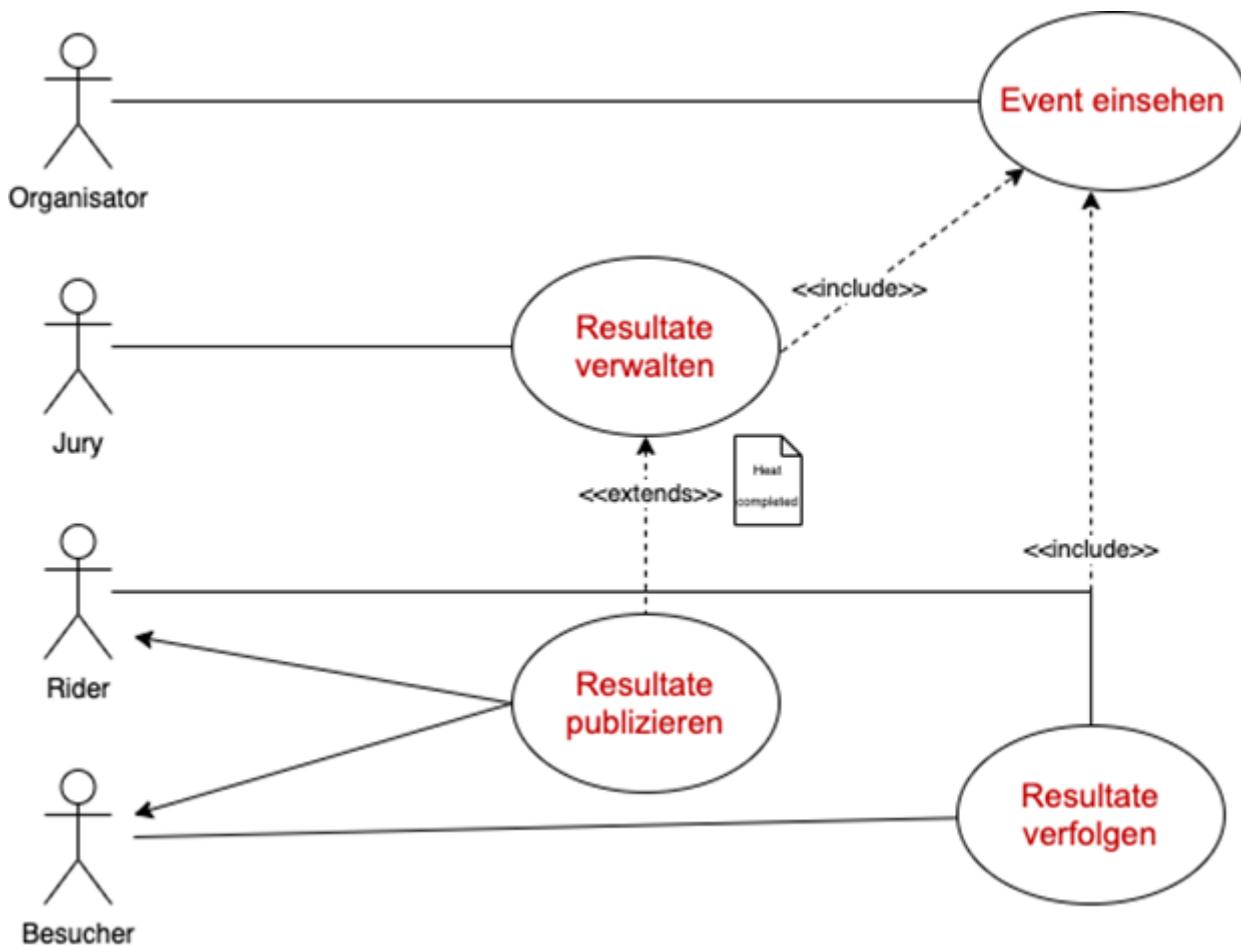
Wir nehmen die wachsende Popularität des Sports zum Anlass, um eine schweizweite App für die Planung, Organisation und Buchführung von Fluss-Surf-Events zu lancieren. Die Applikation hat diverse Zielgruppen, welche unterschiedliche Bedürfnisse stellen. Diese Bedürfnisse werden je nach Kontext Mobile- oder Desktop-First realisiert.

- MVP = Soll im "Minimum Viable Product" umgesetzt werden.
- Nice to have = Wird bei günstigem Projektverlauf umgesetzt.
- Mock = Wird gemockt umgesetzt

Inhalt

[[TOC]]

Rollen und Bedürfnisse



Besucher

Resultate verfolgen (MVP)

Der Besucher möchte Resultate des Surf-Event verfolgen.

- Kontext: Zuhause (Desktop)
- Fokus: Mobile-First

Auf dem Laufenden über den Surf-Event sein (MVP)

Der Besucher möchte laufend Informationen zum Surf-Event erhalten (Aktuelle Rundeninformationen, Teilnehmende Riders)

- Kontext: Vor Ort
- Fokus: Mobile-First

Sich über einen Surf-Event informieren (MVP)

Der Besucher möchte Informationen zum Surf-Event erhalten (Nächster Surf-Event, Start- & Enddatum, Austragungsort).

- Kontext: Zuhause (Desktop) und vor Ort
- Fokus: Mobile-First

Verfolgen eines bestimmten Riders (MVP)

Der Besucher möchte die Tätigkeit eines Riders verfolgen und bei Neuigkeiten informiert werden.

- Kontext: Vor Ort
- Fokus: Mobile-First

Bilder vom Surf-Event posten und ansehen (Nice to have)

Der Besucher möchte Bilder des Surf-Events posten und weitere Bilder des Anlasses betrachten können (Feed).

- Kontext: Vor Ort
- Fokus: Mobile-First

Daten über Aare und Wetter ansehen (Nice to have)

Der Besucher möchte die aktuellen Rahmenbedingungen (Temperatur, Wetter) des Surf-Events erkennen.

- Kontext: Vor Ort
- Fokus: Mobile-First

Rider

Registrierung (Mock)

Der Rider soll sich auf der Plattform registrieren können.

Mock: Die Rider sind bereits vorgängig in der Datenbank erfasst

Anmeldung an einem Surf-Event (Mock)

Der Rider soll sich für ein Surf-Event anmelden können.

Mock: Die Rider sind bereits vorgängig für Surf-Events angemeldet

Eigene Profilseite verwalten (Nice to have)

Der Rider kann seine Profilseite bearbeiten (Foto, Beschreibung, Kontakt).

Resultate verfolgen (MVP)

Der Rider möchte den Verlauf des Surf-Event verfolgen. Er möchte über alle Resultate informiert werden und wissen, welche Runde und Heat anstehen.

- Kontext: Vor Ort
- Fokus: Mobile-First

Über eigene Resultate informiert werden (Nice to have)

Der Rider möchte beim Erhalt eines Resultats selber benachrichtigt werden.

Jury

Surf-Event auswerten (MVP)

Ein Jurymitglied führt die Resultate des Surf-Events nach und teilt die Riders in die verschiedenen Heats ein.

- Kontext: Vor Ort (Desktop)
- Fokus: Desktop-First

Gewinner ermitteln (MVP)

Ein Jurymitglied kann anhand der Resultate die Gewinner der Heats in die nächste Runde übernehmen.

- Kontext: Vor Ort (Desktop)
- Fokus: Desktop-First

Organisator

Surf-Event verwalten (Nice to have)

Ein Organisator kann ein Surf-Event erfassen und bearbeiten.

- Kontext: Zuhause (Desktop)
- Fokus: Desktop-First

Surf-Event einsehen (MVP)

Ein Organisator kann die erfassten Surf-Events einsehen (gemockte Teilnehmerverwaltung, Surf-Event Daten)

- Kontext: Zuhause und vor Ort (Desktop)
- Fokus: Desktop-First

Resultate sollen vom Organisator bestätigt werden (Nice to have)

Zur Absicherung der Resultateingabe sollen die von der Jury eingegebenen Resultate durch den Organisator bestätigt werden.

- Kontext: Zuhause und vor Ort (Desktop)
- Fokus: Desktop-First

Teilnehmerverwaltung (Mock)

Der Organisator kann registrierte Riders einem Surf-Event zuweisen oder dies entfernen.

Mock: Die Rider sind vorgängig bereits den Surf-Events zugewiesen und können nicht entfernt werden.

Export (Nice to have)

Der Organisator soll ein Export des Surf-Events erstellen können, in welchem alle Daten des Surf-Events und die Resultate hinterlegt sind.

Weiter zu: Scope des Projektes

Scope des Projektes

[[TOC]]

Surf-Event Infos sehen

Vorgabe (MVP)

Jeder Besucher kann die Basisinformationen eines Surf-Events (Titel, Start, Ende, Art der Wettkämpfe, Teilnehmer) ansehen.

Mock: Die einzelnen Surf-Events werden dabei als Surf-Event gemockt

Umsetzung

Beim Aufruf der Seite kann der Benutzer direkt erfahren, welche Surf-Events aktuell oder demnächst anstehen. Auch vergangene Surf-Events können im Reiter unterhalb von "Check out our Jams" ausgewählt werden.

The screenshot shows the RiverSurf website interface. At the top, there's a navigation bar with a logo, a search bar, and a bell icon. Below it, a main heading says "Check out our Jams". Underneath, there are three tabs: "Latest Jams" (which is active), "Upcoming Jams", and "Past Jams". A red arrow points to the "Past Jams" tab. The main content area displays three event cards:

- RiverSurf Marzili 2021**: Located in Marzili, Bern on 09.10.2021. Description: "Das ist die Beschreibung vom Event".
- Kappelbridge Jam 2021**: Located at Kappelbrücke, Luzern on 16.10.2021. Description: "Das ist die Beschreibung vom Event".
- RiversurfJam Thun 2022**: Located at Welle Mühlenschleuse, Thun on 10.09.2022. Description: "Das ist die Beschreibung vom Event".

To the right, there's another section titled "Check out our awesome Riders" which lists several users with their roles and icons.

Die Surf-Events werden als Karte dargestellt. Dabei befindet sich neben dem Logo, einem Titel, dem Austragungsort, einem Bild und einer Beschreibung des Surf-Events auch gleich die "Competitions", also die eigentlichen Wettkämpfe. Im untenstehenden Beispiel gibt es jeweils einen Wettkampf für Männer und für Frauen.

 RiversurfJam Thun 2021
Welle Mühleschleuse, Thun
11.09.2021



Das ist die Beschreibung vom Event

[Male](#) [Female](#)

Klickt der Benutzer auf den Surf-Event, so landet er auf der Detailseite des Surf-Events. Hier werden links auch die Start- und Endzeiten dargestellt.

Mittig wird ein Foto des Surf-Events angezeigt, rechts sieht man die teilnehmenden Riders und unten befindet sich eine Karte mit dem Standort des Surf-Events.



RiversurfJam Thun 2021

Description
Das ist die Beschreibung vom Event

Start
11 September 2021 09:00

End
11 September 2021 22:00

Location
Welle Mühlenschleuse, Thun

Hashtag
#riversurf-thun

**Competitions****Male****Female****Riders in Action**

Surf-Event Resultate sehen

Vorgabe (MVP)

Jeder Besucher kann die Resultate des Surf-Events (Punktergebnisse, Runden, Gewinner) betrachten.

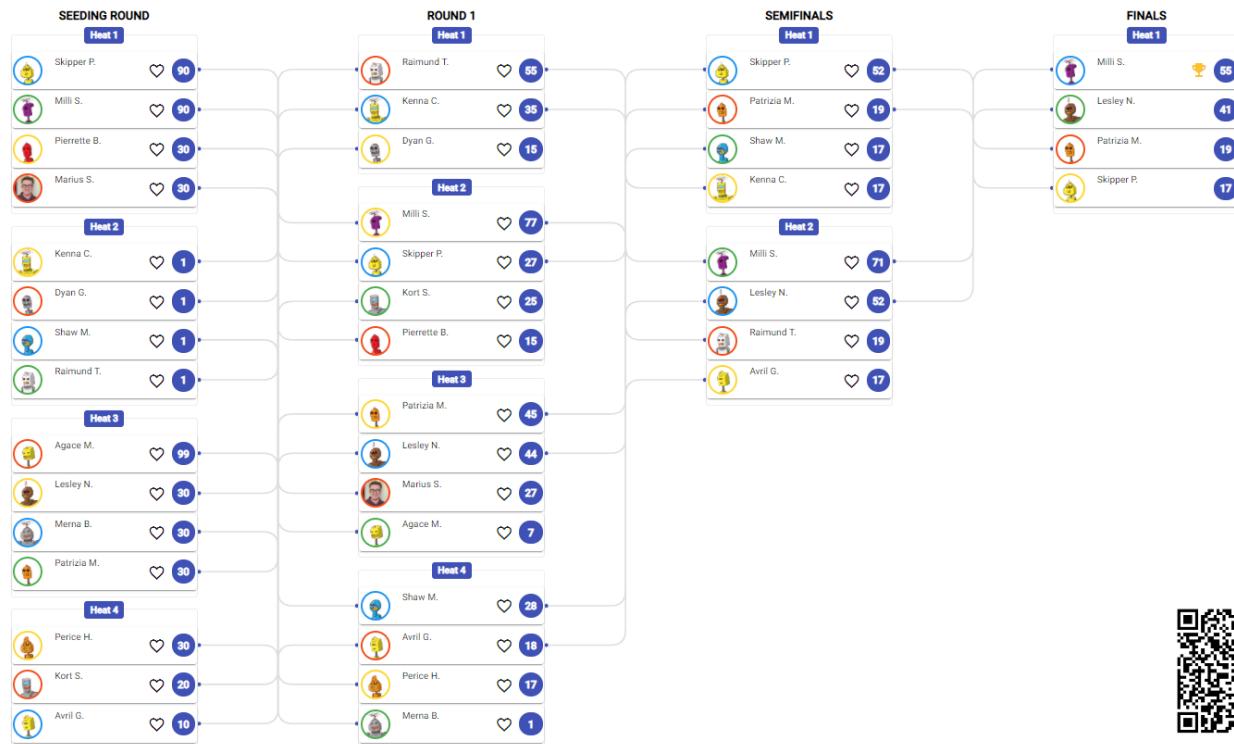
Umsetzung

Befindet sich der Benutzer auf dem Surf-Event, so kann er (sofern eine Competition für den Surf-Event vorhanden ist) auf die Competition klicken, um den aktuellen Stand des Wettbewerbs zu betrachten.

Competitions

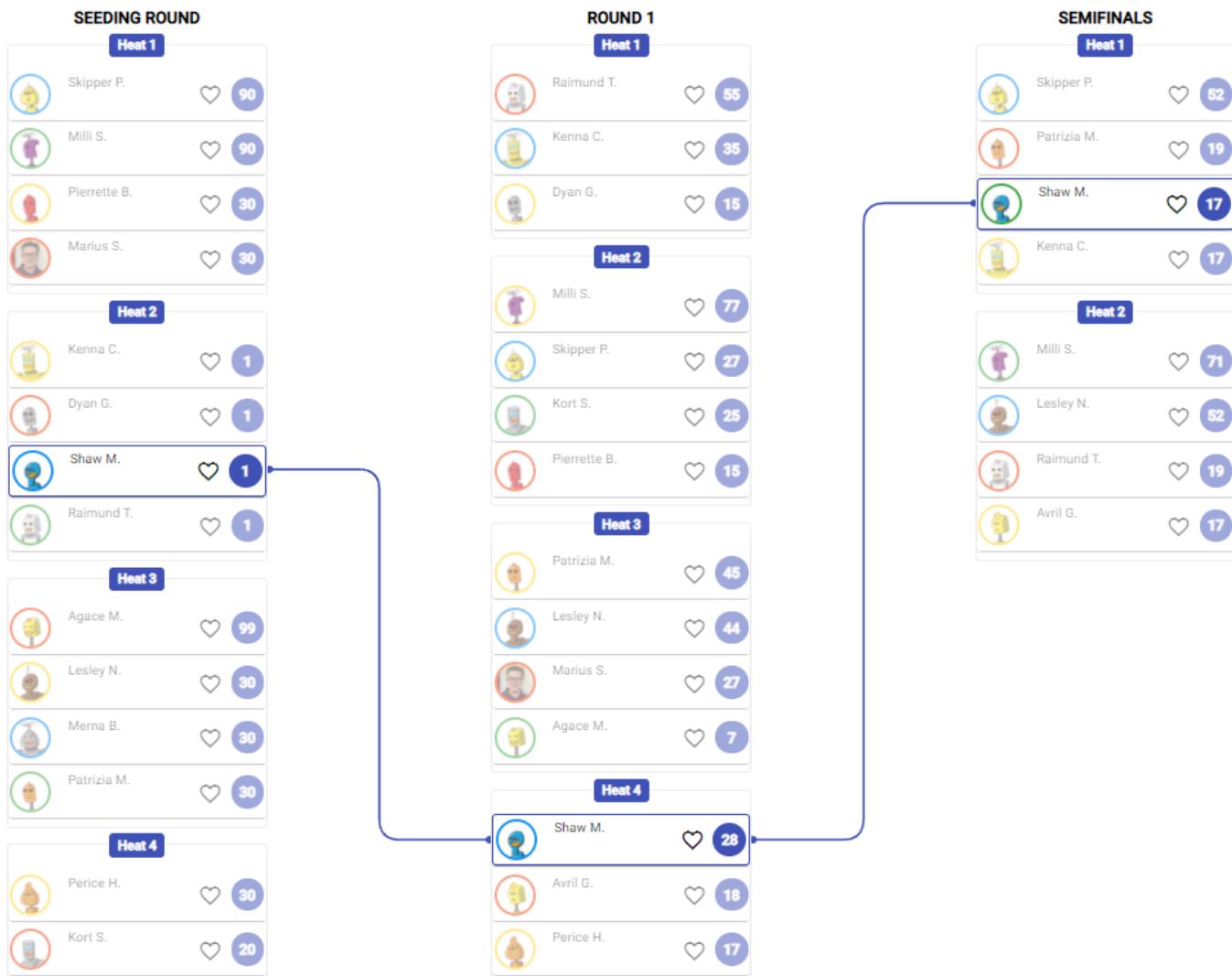
Male**Female**

Die Darstellung des Wettkamps zeigt jeweils von links nach rechts die Runden und dazugehörigen Heats an. Oben rechts befindet sich ebenfalls die Möglichkeit zu einem anderen Wettkampf desselben Surf-Events zu gelangen.

RiversurfJam Thun 2021
♂ Male
♀ Female


Unten rechts befindet sich ausserdem ein QR-Code, welcher den direkten Link zu der dargestellten Seite beinhaltet. Die Idee dahinter ist, dass diese Wettkampfdarstellung während eines Surf-Events auf mehreren Screens angezeigt wird. Mittels des QR-Codes können Besucher diese Anzeige direkt auf dem Smartphone aufrufen und so den weiteren Verlauf selbständig verfolgen.

Klickt der Besucher auf einen Rider, so wird sein Turnierverlauf hervorgehoben.



Erhält Benachrichtigungen über Surf-Event

Vorgabe (MVP)

Ein Besucher erhält eine Benachrichtigung, wenn ein neuer Heat startet oder neue Resultate vorliegen.

Umsetzung

Der Grundgedanke dieser Funktion war es, dass ein Benutzer neben einem Rider auch einem Surf-Event "folgen" kann. Die Funktion des verfolgen eines Surf-Events wurde aber nicht umgesetzt. Durch die Favorisierung eines Riders erhält der Benutzer aber dennoch Benachrichtigungen zum Surf-Event, insofern der Rider an diesem teilnimmt.

Fotos vom Surf-Event betrachten

Vorgabe (Nice to have)

Der Besucher kann aktuell gepostete Medien (Social Media, Instagram) zum Surf-Event betrachten. Diese werden mit einem vordefinierten Hashtag zum Surf-Event auf Instagram publiziert und ausgelesen.

Umsetzung

Dieses Feature wurde nicht umgesetzt.

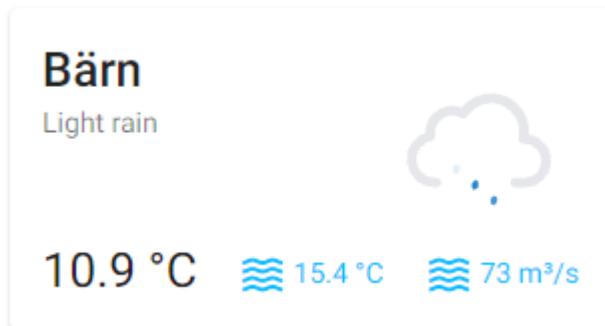
Es werden zusätzliche Informationen zum Surf-Event bereitgestellt

Vorgabe (Nice to have)

Über 3rd Party-APIs werden Daten zum Wasser visualisiert (Aare.guru API: Wassertemperatur, Lufttemperatur, Fliessgeschwindigkeit).

Umsetzung

In der Wettkampf-Ansicht befindet sich unten rechts neben dem QR-Code auch die aktuellen Wetterkonditionen mit Luft- und Wassertemperatur sowie Strömung des Austragungsortes.



Da die verwendete 3rd Party-API nicht alle möglichen Wettkampfstandorte abdeckt, versuchen wir den Austragungsort einer vorhandenen Messstation zuzuordnen. Falls dies nicht möglich ist, werden die Zusatzdaten nicht angezeigt.

Weiter zu: [Funktionen als Rider](Scope des Projektes/Funktionen als Rider)

[[TOC]]

Kann sich auf Website registrieren

Vorgabe (Mock)

Ein Rider kann sich auf der Website neu registrieren.

Mock: Die Rider erhalten ein vorgegebenes Login

Kann sich für ein Surf-Event anmelden

Vorgabe (Mock)

Ein Rider kann sich für ein Surf-Event anmelden.

Mock: Die Rider sind bereits bei Surf-Events registriert

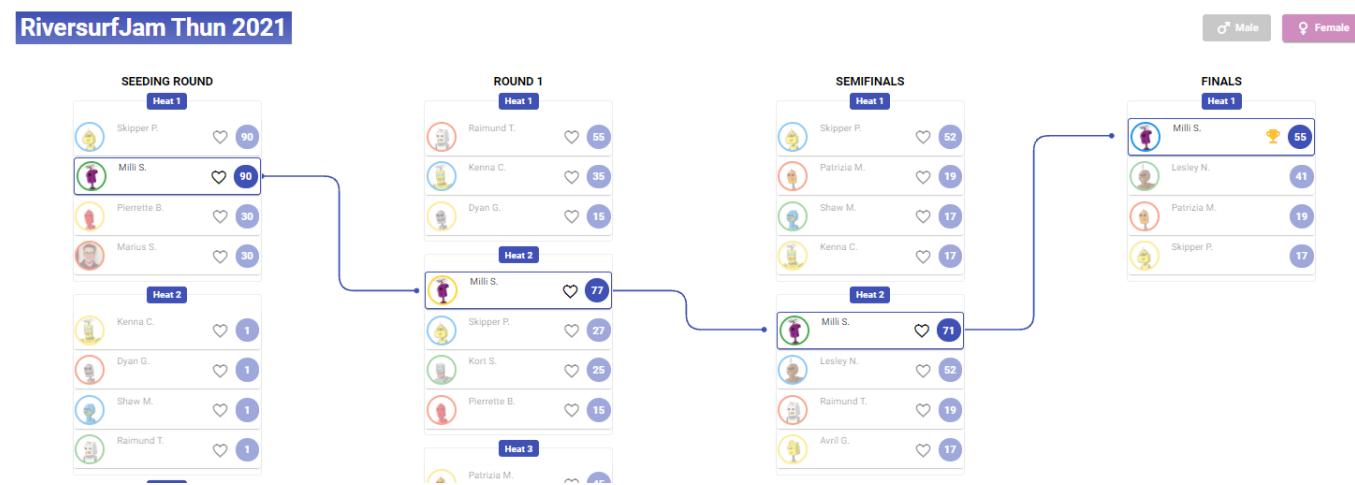
Will seine Resultate einsehen

Vorgabe (MVP)

Der Rider kann seine eigenen Resultate (Punkte, Rangliste) einsehen.

Umsetzung

Der Rider kann gleich wie der Besucher zum Surf-Event navigieren und auf der Ansicht der Competition die Resultate einsehen.



Und auf seinem Profil die vergangenen Aktivitäten betrachten.

The screenshot shows a mobile application interface. At the top, there are social media icons for Facebook, Instagram, and Twitter. Below them is a circular profile picture of a pink robot-like character. Underneath the profile picture is a small heart icon. The rider's name, "Ermentrude Osborn", is displayed in large blue text, followed by "ability ♀". Below the name is a quote: "«Optional optimizing monitoring»". At the bottom, it says "Born: 1 February 1998 (age 23)".

2020

RiversurfJam Thun 2020

WELLE MÜHLESCHLEUSE, THUN

Das ist die Beschreibung vom VERGANGENEN Event

- Participates in Heat 1 of round 0** ▶
09:16 | 6. October 2021
- Completed Heat 1 of round 0** ▣
09:16 | 6. October 2021
- Was rated in Heat 1 of round 0** ▣
09:16 | 6. October 2021
Scored 30 points
- Participates in Heat 0 of round 1** ▶
09:17 | 6. October 2021
- Completed Heat 0 of round 1** ▣
09:17 | 6. October 2021
- Was rated in Heat 0 of round 1** ▣
09:17 | 6. October 2021
Scored 20 points

Will über seine Resultate benachrichtigt werden

Vorgabe (Nice to have)

Der Rider wird bei der Publikation von Resultaten benachrichtigt.

Umsetzung

Dieses Feature wurde nicht umgesetzt, da die Verwaltung des Profils nicht umgesetzt wurde und die Logins vorgegeben sind.

Will sein Profil verwalten können

Vorgabe (Nice to have)

Der Rider kann seine Profildaten anpassen (Name, Vorname, Spitzname, Alter, Surf-Event-Arten, Geburtstag, Mail, Beschreibung, Social-Media).

Umsetzung

Die Profildaten sind fix hinterlegt.



A purple profile card for 'Milli Seager' (matrices ♀). It features a circular icon of a purple robot head with an antenna, a blue background, and social media links for Facebook, Instagram, and Twitter at the top. Below the icon is a small white heart. The name 'Milli Seager' is in large white font, followed by 'matrices ♀' in smaller white font. A quote in white font reads: «Focused foreground open architecture». Below the quote is the birth date 'Born: 22 February 2015 (age 6)'.

Weiter zu: [Funktionen als Jury](Scope des Projektes/Funktionen als Jury)

[[TOC]]

Will Resultate verwalten

Vorgabe (MVP)

Die Jury kann die Resultate des Surf-Events (Punkte, Gewinner, Zuteilung der Rider in die Heats) verwalten.

Umsetzung

Damit der Benutzer mit der Jury-Rolle die Resultate eines Surf-Events verwalten kann, muss sich dieser zunächst anmelden. Dies erfolgt über das Login im Menu.



The screenshot shows the RiverSurf mobile application interface. At the top, there's a blue header bar with the logo 'RiverSurf', a search bar labeled 'Search jams or riders', and a menu icon. Below the header, there's a section titled 'Check out our Jams' with tabs for 'Latest Jams', 'Upcoming Jams', and 'Past Jams'. Underneath are cards for 'RiversurfJam Thun 2021' (Welle Mühleschleuse, Thun, 11.09.2021) and 'RiversurfJam Thun 2020' (Welle Mühleschleuse, Thun, 12.09.2020). To the right, there's a section titled 'Check out our awesome Riders' featuring 'Jerrold Tibbits' (♀ algorithm) with a small profile icon and a heart icon. An upward-pointing red arrow is overlaid on the right side of the rider section.

Nach korrekter Eingabe der Anmeldedaten und erfolgreichen Login erscheint eine Meldung und neu befindet sich oben rechts zwischen dem Notification- und Menu-Icon ein Logout-Knopf.

The screenshot shows the RiverSurf website interface. At the top, there is a blue header bar with the RiverSurf logo, a search bar labeled "Search Jams or riders", and a notification icon. Below the header, a green banner says "YOU'RE LOGGED IN MATE!" with a close button. On the left, there's a sidebar with "Check out our Jams" and links for "Latest Jams", "Upcoming Jams", and "Past Jams". On the right, another sidebar says "Check out our awesome Riders".

Nun kann der Benutzer zum Surf-Event navigieren, die Competition auswählen und in der Ansicht auf "Edit Competition" klicken.

The screenshot shows the competition bracket for "RiversurfJam Thun 2021". The bracket is organized into rounds: Seeding Round, Round 1, Round 2, Semifinals, and Finals. The "Edit Competition" button is highlighted with a red arrow. The Seeding Round shows heats 1 and 2 with rider names and scores. The Round 1 bracket shows four heats: Heat 1 (Luciano D., Roanne M., Arny S., Kimball I.), Heat 2 (Jerrold T., Kimball I.), Heat 3 (Gerick A., Creighton G.), and Heat 4 (Clarie O., Luciano D.). The Round 2 bracket shows four heats: Heat 1 (Creighton G., Gerick A., Normand L., Roanne M.), Heat 2 (Clarie O., Luciano D.), Heat 3 (Dun Satterly, Doralin Ormiston), and Heat 4 (Gillie Salzberg, Raimund Tesdale). The Semifinals and Finals sections are currently empty.

So landet das Jury-Mitglied auf der aktuellen Runde und sieht die Heats in welchen die einzelnen Rider gegeneinander antreten.

The screenshot shows the "Round 2" page with three heats listed: Heat 1, Heat 2, and Heat 3. Heat 1 has been completed ("Save Heat" button is active) and has 4 / 4 entries. Heat 2 is currently running ("End Heat" button is active) and has 2 / 4 entries. Heat 3 has not started yet ("Save Heat" button is inactive) and has 4 / 4 entries. Each heat lists the rider names and their corresponding points input fields.

| Heat | Rider Name | Points |
|--------|----------------------|----------------------|
| Heat 1 | Creighton Greensides | <input type="text"/> |
| Heat 1 | Gerick Artharg | <input type="text"/> |
| Heat 1 | Normand Lardiner | <input type="text"/> |
| Heat 1 | Roanne Moxom | <input type="text"/> |
| Heat 2 | Clarie O'Nion | <input type="text"/> |
| Heat 2 | Luciano Dy | <input type="text"/> |
| Heat 3 | Dun Satterly | <input type="text"/> |
| Heat 3 | Doralin Ormiston | <input type="text"/> |
| Heat 3 | Gillie Salzberg | <input type="text"/> |
| Heat 3 | Raimund Tesdale | <input type="text"/> |

Heat 1 und 3 sind bereits beendet, Heat 2 ist noch am laufen. Die beendeten Heats warten nun auf die Punkteeingabe durch den Benutzer. Werden alle Punkte in einem Heat eingegeben, schaltet sich der Button "Save Heat" frei und die Resultate werden in der Competition-Ansicht nachgeführt.

| Heat 1 | Heat 2 |
|------------------------------------|------------------------------------|
| Creighton Greensides Points: 23 | Creighton Greensides Points: 23 |
| Gerick Artharg Points: 17 | Gerick Artharg Points: 17 |
| Normand Lardiner Points: 16 | Normand Lardiner Points: 16 |
| Roanne Moxom Points: 22 | Roanne Moxom Points: 22 |

Falls sich Benutzer bei einem Rider für Neuigkeiten registriert haben, erhalten diese nach dem Speichern der Resultate eine entsprechende Benachrichtigung.

Wurden alle Resultate der Runde erfasst und gespeichert, so schaltet sich der Button "Move to next round" frei, welcher alle Sieger der Heats eine Runde weiterschiebt.

| Heats | | |
|--|---|--|
| Heat 1 Save Heat ✓ Creighton Greensides Points: 23 | Heat 2 Save Heat ✓ Clarie O'Nion Points: 42 | Heat 3 Save Heat ✓ Dun Satterly Points: 12 |

Es wird automatisch in die neue Runde gewechselt. Hier kann der Benutzer mit der Jury-Rolle nun die Rider in die Heats per Drag & Drop zuteilen oder über den Button "Assign Riders Randomly" zufällig verteilen.

| Unassigned riders | | Heats | |
|-------------------|--|-------|--|
| Creighton G. X | Heat 1 Luciano D. Roanne M. X | | |
| Raimund T. X | Heat 2 Clarie O. X | | |
| Gillie S. X | | | |

Sind alle Rider verteilt, können die einzelnen Heats erneut gestartet werden.

The screenshot shows a software interface for managing surf events. At the top, there are buttons for "Assign Riders Randomly", "Revert", and "Move to next round". Below this, a section titled "Heats" is shown. Heat 1 contains four entries: Luciano Dy, Roanne Moxom, Creighton Greensides, and Raimund Tesdale, each with a thumbs-up icon. Heat 2 contains two entries: Clarie O'Nion and Gillie Salzberg, also each with a thumbs-up icon.

| Heat | Rider | Action |
|--------|----------------------|-----------|
| Heat 1 | Luciano Dy | Thumbs Up |
| Heat 1 | Roanne Moxom | Thumbs Up |
| Heat 1 | Creighton Greensides | Thumbs Up |
| Heat 1 | Raimund Tesdale | Thumbs Up |
| Heat 2 | Clarie O'Nion | Thumbs Up |
| Heat 2 | Gillie Salzberg | Thumbs Up |

Weiter zu: [Funktionen als Organisator](Scope des Projektes/Funktionen als Organisator)

[[TOC]]

Will zwischen Surf-Events auswählen können

Vorgabe (Mock)

Der Organisator kann zwischen den erstellten Surf-Events auswählen.

Mock: Es gibt nur die vordefinierten Surf-Events

Umsetzung

Die Verwaltung der Surf-Events wurde nicht umgesetzt.

Will Surf-Events verwalten können

Vorgabe (Nice to have)

Der Organisator kann die Daten des Surf-Events verändern.

Mock: Es gibt nur die vordefinierten Surf-Events, welche nicht gelöscht oder verändert werden können.

Umsetzung

Die Verwaltung der Surf-Events wurde nicht umgesetzt.

Kann die Teilnehmer eines Surf-Events verwalten

Vorgabe (Mock)

Der Organisator kann die angemeldeten Rider zum Surf-Event zulassen.

Mock: Die Rider sind bereits zum Surf-Event zugelassen.

Umsetzung

Die Verwaltung der Surf-Events wurde nicht umgesetzt.

Bestätigung der Resultate

Vorgabe (Nice to have)

Die von der Jury eingegebenen Resultate müssen vom Organisator bestätigt werden und erst anschliessend publiziert.

Umsetzung

Die Bestätigung durch den Organisator wurde nicht umgesetzt. Resultate werden direkt vom Benutzer mit der Jury-Rolle publiziert.

Export der Resultate

Vorgabe (Nice to have)

Die Resultate eines Surf-Events (Teilnehmerliste, Rangliste, Turnierresultate) können zur Archivierung exportiert werden.

Umsetzung

Die Exportfunktion wurde nicht umgesetzt.

Weiter zu: [Generelle Funktionen](Scope des Projektes/Generelle Funktionen)

[[TOC]]

Offlinefähigkeit für Resultateingabe

Vorgabe (MVP)

Die Funktionalität soll auch offline angeboten werden, da eine Auswertung des Wettbewerbs auch bei einem Ausfall des Internets garantiert werden soll. Davon ausgenommen sind Zugriffe auf 3rd-Party-APIs (z.B. Wetter-Daten).

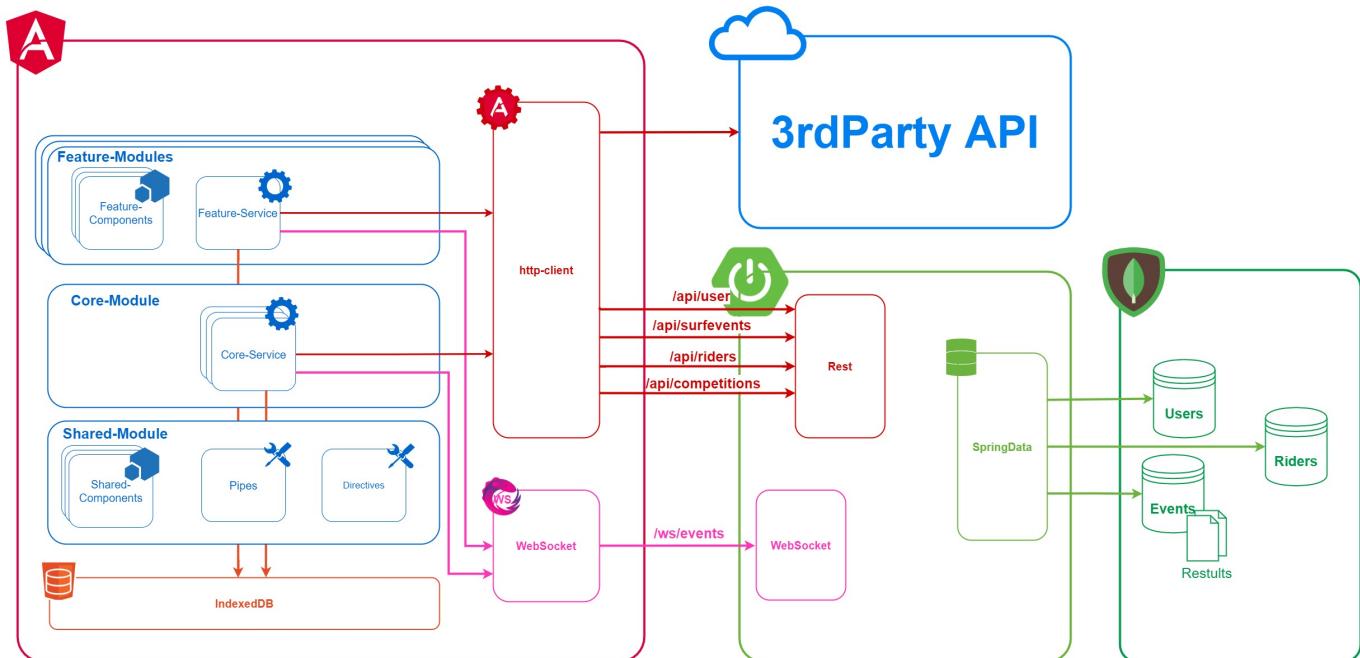
Umsetzung

Die Offlinefunktionalität für die Resultateingabe wurde über zwei Wege erreicht. Einerseits wurde die PWA-Package verwendet, welche es erlaubt, Requests und Daten zu Cachen und andererseits wurden geladene Daten und zu sendende Daten in einer IndexedDB abgespeichert.

Beim erstmaligen Laden der App werden neben den Daten der Rider auch alle aktuell laufenden Surf-Events und die darin enthaltenen Wettbewerbe geladen. Tritt nun der Fall ein, dass die Anwendung nun offline läuft, werden die Änderungen am Wettbewerb einerseits seitens PWA gespeichert (REST-Zugriff auf Competition-Endpoint) und die Benachrichtigungen (Starten eines Heats, Stoppen, Resultateingabe) in der IndexedDB. Die Benachrichtigungen werden dann beim erneuten Eintritt in den Onlinemodus per WebSocket abgeschickt und die zwischengespeicherten Anpassungen am Wettbewerb gepushed.

Weiter zu: [Systemübersicht](#)

Systemübersicht



Inhalt

[[TOC]]

Angular Anwendung

App Module

Im App Module befindet sich das Bootstrapping sowie die Shell.

Core Module

Als Core Module werden diejenigen Komponenten bezeichnet, welche in der gesamten Anwendung verwendet werden sowie Komponenten, welche in der Shell der Applikation verwendet werden.

Komponenten

- HomeComponent, welche die Startseite mit den Surf-Events, einer Liste mit zufälligen Riders und eine Karte mit den Surf-Events als Marker beinhaltet

The screenshot shows the RiverSurf website interface. At the top, there is a header bar with the logo "RiverSurf" and a search bar labeled "Search jams or riders". Below the header, there is a red box highlighting the "Check out our Jams" section. This section contains three tabs: "Latest Jams" (selected), "Upcoming Jams", and "Past Jams". Under the "Latest Jams" tab, there is a card for "Oakwood Jam 2021" in Eichholz, Bern, on 19.09.2021. The card includes a photo of people surfing and a placeholder text "Das ist die Beschreibung vom Event". To the right of the "Jams" section, there is another red box highlighting the "Check out our awesome Riders" section, which lists six rider profiles with their names, gender, and roles.

| Rider Name | Gender | Role |
|----------------|--------|-----------------|
| Vale Easthope | ♂ | approach |
| Lila Idney | ♂ | info-mediaries |
| Verla Chetwynd | ♀ | user-facing |
| Jaimie Borrott | ♀ | help-desk |
| Appolonia Sear | ♂ | value-added |
| Dyan Guilleton | ♂ | instruction set |

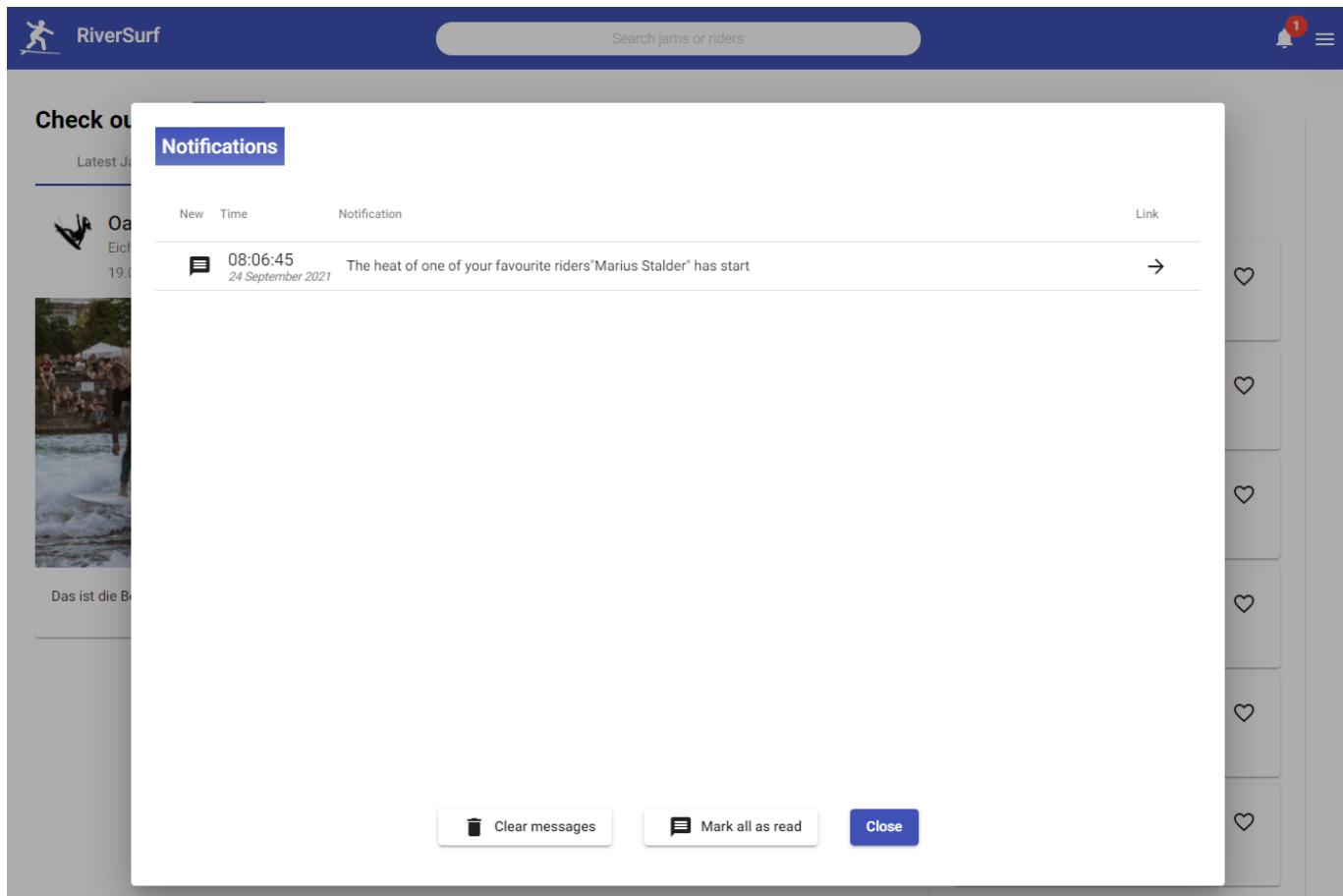
- HeaderComponent, welche die Suche sowie die Navigation beinhaltet

The screenshot shows the RiverSurf website's navigation bar, which includes the "RiverSurf" logo, a search bar, and notification icons (bell and menu).

- NavigationComponent, welche die Benachrichtigungskomponente sowie das Menu beinhaltet



- NotificationComponent, die dem Benutzer Benachrichtigungen anzeigen kann



Services

Hier eine nicht abschliessende Liste der Services im Core Modul:

- UserService, wird für das Login und die favorisierten Riders verwendet
- RidersService, wird für die Abfrage von Riders verwendet
- SurfEventService, wird für die Abfrage von Surf-Events, dazugehörigen Competitions und Riders sowie das Aktualisieren einer Competition des Surf-Events verwendet
- WebSocketService, wird für den Aufbau einer Websocket-Verbindung zum Server und dem dazugehörigen Datenaustausch verwendet
- SearchService, ist für die Suche nach Riders sowie Surf-Events in der Anwendung zuständig
- NetworkStatusService, zur Erkennung des Netzwerkstatus (Online/Offline)
- SnackbarService, zur Darstellung von Mitteilungen auf dem Bildschirm
- UserNotificationService, zum Erhalt und Versand von Benachrichtigungen
- RouterHistoryService, um die Routen des Benutzers abzuspeichern, wird im ErrorComponent verwendet

Models

Hier werden in der Anwendung verwendete Datenmodelle nicht abschliessend aufgelistet. Diese sind jeweils als Interface exportiert.

- User, Datenstruktur für einen Benutzer

```
export interface User {
  id: string;
  userName?: string;
```

```
email?: string;
userRole?: Role;
profile?: string;
token?: string;
isAuthenticated?: boolean;
favouriteRiders: string[];
}
```

- SurfEvent, Datenstruktur für ein Surf-Event

```
export interface SurfEvent {
  id: string;
  name: string;
  description: string;
  logo: string;
  mainPicture: string;
  startDateTime: Date;
  endDateTime: Date;
  location: string;
  locationLat: number;
  locationLong: number;
  hashTag: string;
  competitions: string[];
  judge: string;
  organizer: string;
  divisions: Division[];
}
```

- Competition, Datenstruktur für einen Wettkampf an einem Surf-Event (und dazugehörige Datenstrukturen)

```
export interface Competition {
  id: string;
  division: Division;
  config: CompetitionConfig;
  riders: string[];
  rounds: Round[];
}

export interface CompetitionConfig {
  maxRiders: number;
  maxRidersInHeat: number;
  winnersInHeat: number;
}

export interface Round {
  id: number;
  riders: string[];
  heats: Heat[];
}
```

```
}
```

```
export type HeatState = 'idle' | 'running' | 'finished' | 'completed';
```

```
export interface Heat {
  id: number;
  riders: string[];
  state: HeatState;
  results: Result[];
}
```

```
export interface Result {
  id?: number;
  riderId: string;
  color: number;
  value: number;
}
```

Shared Module

Komponenten

- CarouselComponent, zur Darstellung von unterschiedlichem Content als Karussell, welches in der Mobilien-Ansicht auch per Touch bedienbar ist ([weitere Infos zur Karussell-Komponente](Eingesetzte Technologien#eigene-karussell-komponente))

Check out our Jams

< Current Next Previous >

 RiversurfJam Thun
2021

Welle Mühleschleuse, Thun
11.09.2021



Das ist die Beschreibung vom Event

Male Female

- EventCardComponent, als Komponente für die Auswahl eines Surf-Events

Check out our Jams

[Latest Jams](#)[Upcoming Jams](#)[Past Jams](#)

RiversurfJam Thun 2021

Welle Mühleschleuse, Thun

11.09.2021



Das ist die Beschreibung vom Event

[Male](#)[Female](#)

- RiderCardComponent, als Komponente zur Darstellung eines Riders

Check out our awesome Riders

Milena Alejandre
♂ orchestration

Doria Gotter
♂ Persistent

Moise Fathers
♂ Innovative

- FavoriteRiderComponent, als Button mit der Funktionalität zum favorisieren eines Riders

Jaimie Borrott
help-desk ♀

« Balanced 5th generation hierarchy »

Born: 28 January 1999 (age 22)

Check out our awesome Riders

Agace MacGebenay
♂ Diverse

Lilyan Andrault
♀ migration

Alley McCrachen
♂ eco-centric

- ErrorComponent, als Fehlerkomponente, welche dargestellt wird, wenn keine passende Route gefunden wurde

There is no **wave** to ride here...

We're sorry mate, but the requested resource "**se**" could not be found.



Did you mean the following **Riders** :

Pierrette Baltrushaitis (instruction set)

Milli Seager (matrices)

Dyan Guilletton (instruction set)

Agace MacGebenay (Diverse)

Rosalie Scuse (product)

Byrom Grelak (Graphical User Interface)

Jacky Gianullo (Enterprise-wide)

Dorise Borborough (client-driven)

Kaiser Pideon (architecture)

Did you mean the following **Jams** :

RiversurfJam Thun 2021

RiversurfJam Thun 2022

RiversurfJam Thun 2020

Your last **destinations**:

se

Pipes

Im Shared-Modul wurden auch Pipes abgelegt:

- DivisionColorPipe, um die Farbe der Wettkämpfe zu erhalten (Male / Female / Kid)
- RiderColorPipe, um die Farbe eines Riders im Heat zu erhalten
- SlugifyPipe, um aus Text einen gültigen URL-Parameter zu formen (SEO). So werden Umlaute, Sonderzeichen und Leerzeichen ersetzt.

Feature-Modul User

Komponenten

- LoginComponent, zur Verwaltung des Logins und der Registrierung

Login Register





- SignupFormComponent, für die Dateneingabe vom Login oder der Registrierung (zwei Ansichten)

Login Register













Feature-Modul Riders

Komponenten

- RidersComponent, Komponente für die Darstellung der vorhandenen Riders mit Filter und Pagination



Browse or Search for Riders

Search Riders

| Name | Nickname | Division |
|-------------------------|-----------------|----------|
| Baltrushaitis Pierrette | instruction set | ♀ Female |
| Stalder Marius | magi | ♂ Male |
| Pluvier Skipper | strategy | ♂ Male |
| Seager Milli | matrices | ♀ Female |
| Chessill Kenna | throughput | ♂ Male |
| Guilleton Dyan | instruction set | ♂ Male |

Your ❤️ Riders:

| | | |
|-----------------|---------------|----|
| Alley McCracken | ♂ eco-centric | ❤️ |
| Lilyan Andrault | ♀ migration | ❤️ |

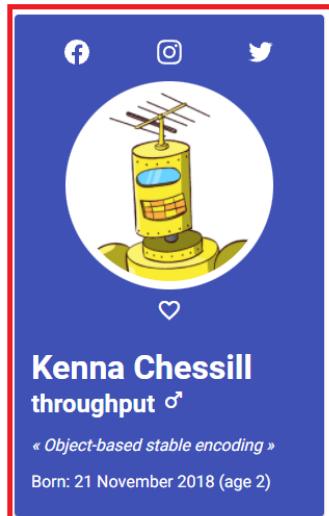
- RiderProfileComponent, für die Darstellung eines einzelnen Rider-Profiles

Kenna Chessill
throughput ♂

« Object-based stable encoding »
Born: 21 November 2018 (age 2)

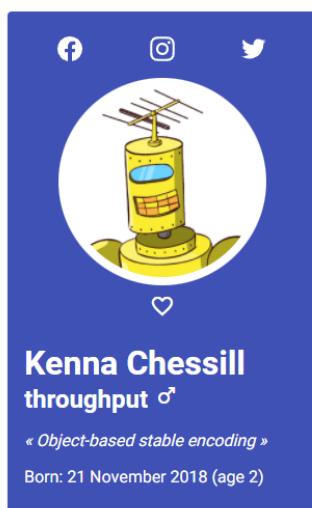
No event data found for the selected rider!
Come back later when the rider has participated in a surf event.

- RiderOverviewComponent, zur Darstellung von Informationen zum Rider



No event data found for the selected rider!
Come back later when the rider has participated in a surf event.

- RiderTimeLineComponent, zur Darstellung von aktuellen und vergangenen Surf-Events an welchen der Rider teilgenommen hat



No event data found for the selected rider!
Come back later when the rider has participated in a surf event.

- TimeLineComponent, zur Darstellung eines Zeitstrahls mit Einträgen

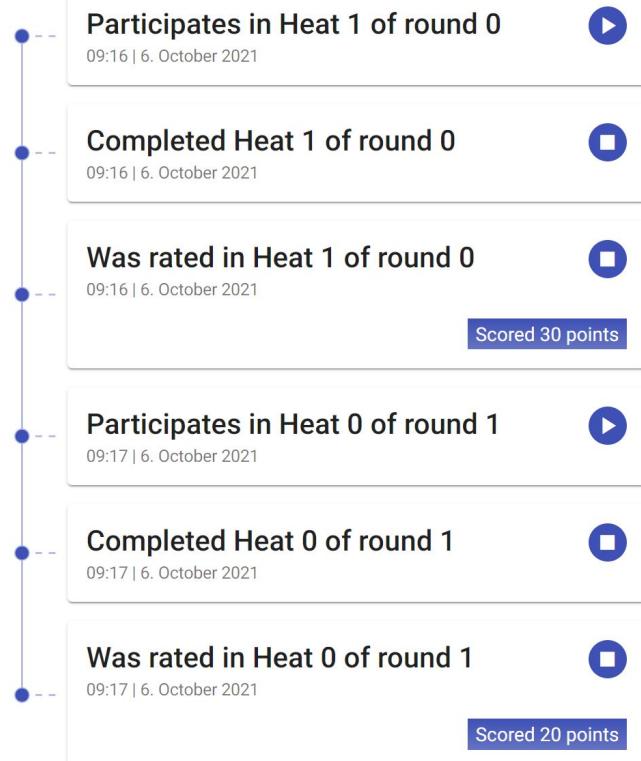
The screenshot shows a user profile with a circular icon of a pink robot head. At the top are social media icons for Facebook, Instagram, and Twitter. Below the icon is a small heart icon. The name "Ermentrude Osborn" is displayed in large white text, followed by "ability ♀". A subtitle "« Optional optimizing monitoring »" is shown in smaller text. At the bottom, it says "Born: 1 February 1998 (age 23)".

2020

RiversurfJam Thun 2020

WELLE MÜHLESCHLEUSE, THUN

Das ist die Beschreibung vom VERGANGENEN Event

**Models**

- EventTimeLine und EventTimeLineCollection als Datenstruktur für einen Zeitstrahl in der TimeLineComponent

```
export interface EventTimeLine {
  id: string;
  event: SurfEvent;
  ongoing: boolean;
  riderId: string;
  timeline: TimeLineItem[];
}

export interface EventTimeLineCollection{
  year: number;
  timeLines: EventTimeLine[];
}
```

- TimeLineItem als Eintrag in einem Zeitstrahl

```
export interface TimeLineItem {
    title: string;
    time: Date;
    content: string;
    icon: TimeLineItemIcon;
}

export type TimeLineItemIcon = 'default' | 'start' | 'finish' | 'lose' | 'win';
```

Services

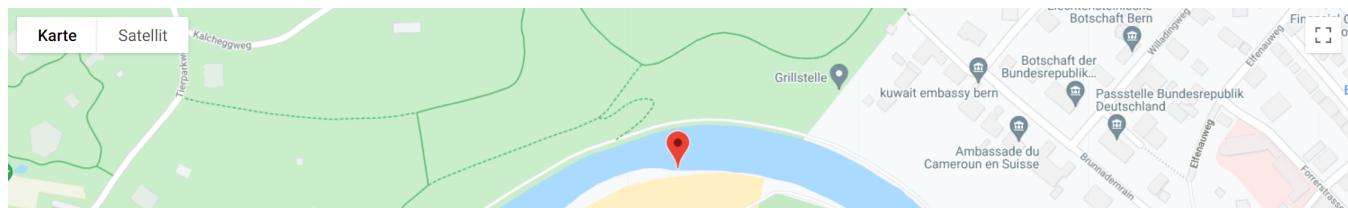
- RiderHistoryService, zur Abfrage von aktuellen und vergangenen Surf-Events, an welchen ein Rider teilgenommen hat

Feature-Modul Surf-Event

Komponenten

- SurfEventComponent, zur Darstellung der Detailinfos zum Surf-Event mit den teilnehmenden Riders und den möglichen Wettkämpfen

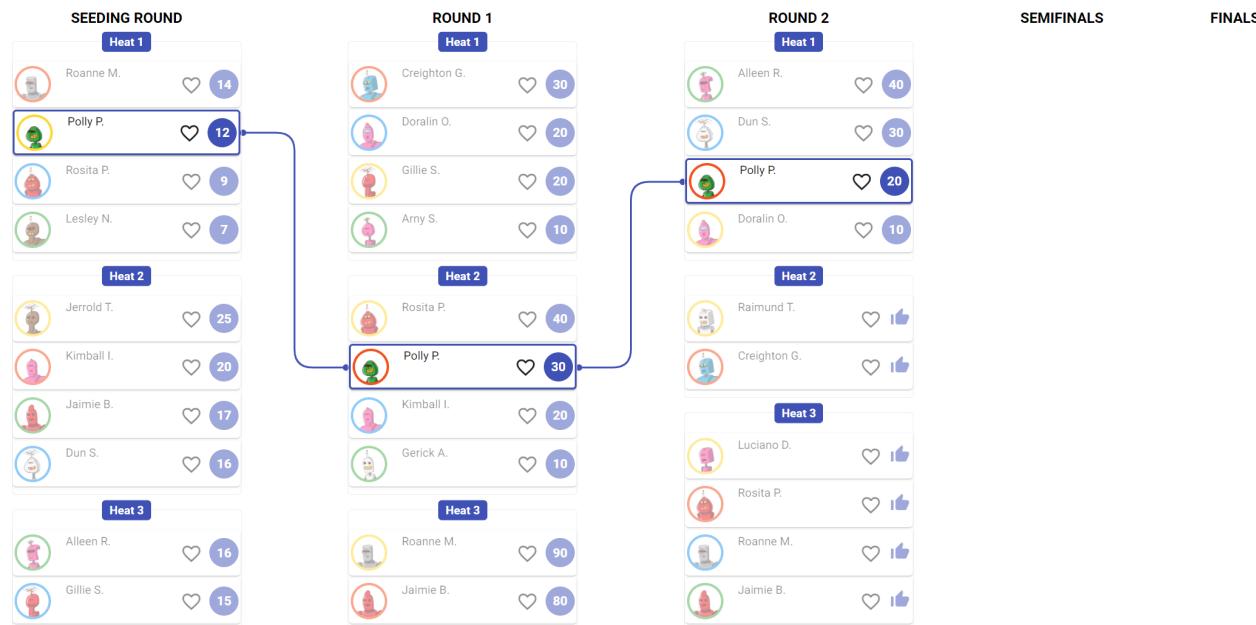
The screenshot shows the RiverSurfJam mobile application interface. On the left, a vertical sidebar displays event details: "Oakwood Jam 2021", "Description" (Das ist die Beschreibung vom Event), "Start" (19 September 2021 10:00), "End" (8 October 2021 18:00), "Location" (Eichholz, Bern), and "Hashtag" (#oakwood-jam). The main area features a large image of three surfers performing a maneuver on a river. To the right, there are two sections: "Competitions" (with Male and Female buttons) and "Riders in Action" (displaying a grid of 24 circular icons, each representing a different rider or competitor).



- CompetitionComponent, zur Bearbeitung und Ansicht eines aktuellen Wettbewerbs vom Surf-Event ([mehr Informationen zur Resultateverwaltung als Jury](Scope des Projektes/Funktionen als Jury#will-resultate-verwalten))

Oakwood Jam 2021

Male Female



- RoundComponent, zur Verwaltung einer Runde in einem Wettbewerb

| Heats | |
|----------------------|-------------|
| Heat 1 | Save Heat ✓ |
| Doralin Ormiston | Points: 10 |
| Polly Pietron | Points: 20 |
| Dun Satterly | Points: 30 |
| Alleen Risen | Points: 40 |
| Heat 2 | Start Heat |
| Raimund Tesdale | 1 |
| Creighton Greensides | 1 |
| Heat 3 | Start Heat |
| Luciano Dy | 1 |
| Rosita Palfy | 1 |
| Roanne Moxom | 1 |
| Jaimie Borrott | 1 |

- HeatComponent, zur Verwaltung eines einzelnen Heats innerhalb einer Runde

Seeding round Round 1 Round 2 Semifinals Finals

Tournament View ♂ Male ♀ Female

Round 2

Move to next round

Heats

| Heat 1 | Heat 2 | Heat 3 |
|----------------------------------|-------------------------------|--------------------------|
| Doralin Ormistonne Points: 10 | Raimund Tesdale Start Heat | Luciano Dy Start Heat |
| Polly Pietron Points: 20 | Creighton Greensides | Rosita Palfy |
| Dun Satterly Points: 30 | | Roanne Moxom |
| Alleen Risen Points: 40 | | Jaimie Borrott |

- RiderResultComponent, zur Darstellung eines Riders in einem Heat, je nach Zustand des Heats, respektive Runde verändert sich das Aussehen der Komponente



- WeatherComponent, zur Darstellung von Wetterdaten zum Ort des Surf-Events

Bärn

Light rain

11.7 °C 15.4 °C 69 m³/s

Services

- WeatherService, zum Erhalt von aktuellen Wetterdaten zu einem Ort eines Surf-Events

State-Management

Der Application-State wird in den jeweiligen Services gehalten und kann von den Komponenten über Observables konsumiert werden. Die Services laden die Daten eigenständig initial oder nach Bedarf.

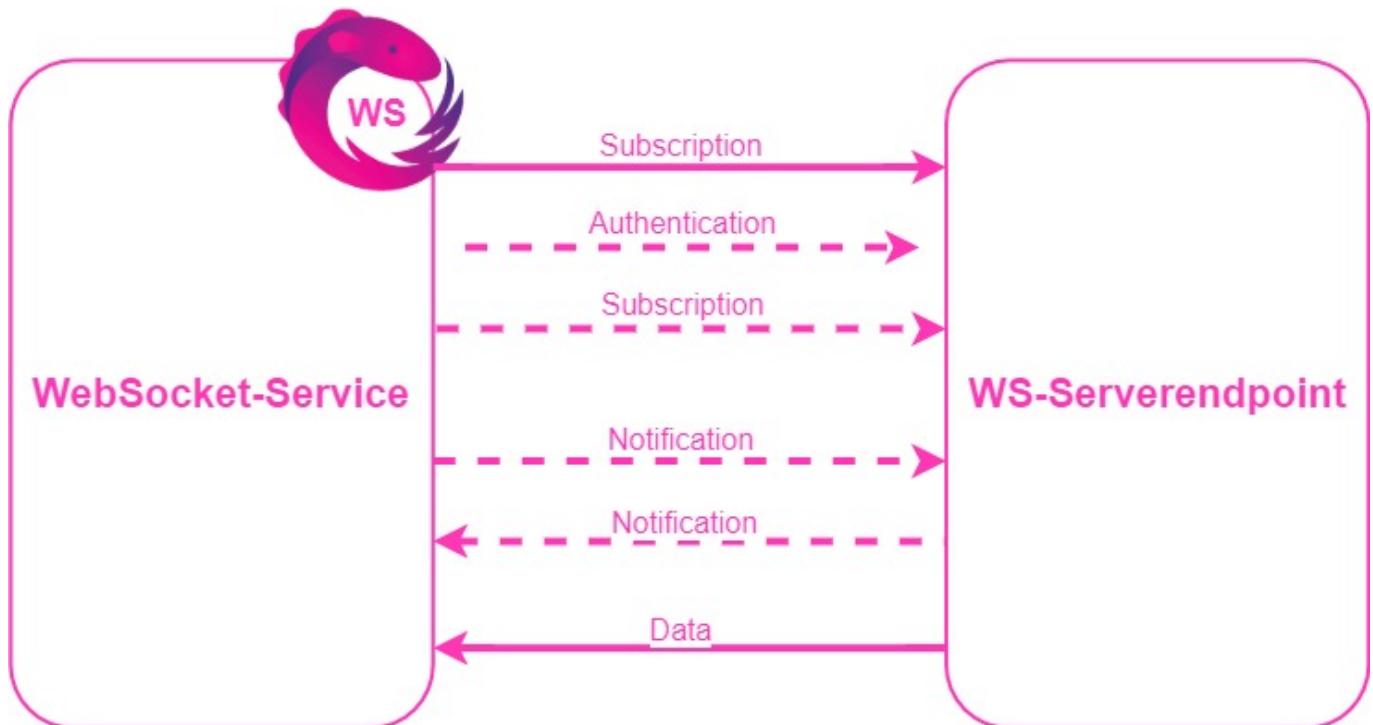
Offline-Mode

Der Erste Ansatz "Offline-first" bei welchem der ganze Application-State in der IndexedDB gehalten und nur im Hintergrund an das Backend übertragen wird, wurde mit dem Einführen der PWA-Funktionalität obsolet. Die bereits implementierte Verwaltung der Riders in der IndexedDB kann in Zukunft ausgebaut werden. Ein anonymer User wird in der IndexedDB angelegt, um auch einem unauthorisierten Benutzer die Möglichkeit zu geben, verschiedenen Riders zu folgen. Ausserdem werden, solange die Applikation offline ist, ausgehende POST und PUT Requests, sowie Websocket Notifications in der IndexedDB abgelegt um sie, sobald wieder eine Netzwerkverbindung zur Verfügung steht, zu Senden und anschliessend aus der IndexedDB zu Löschen. Alle Backend-Requests werden vom ServiceWorker gecached, somit müssen nur Mutationen in der IndexedDB abgelegt und von dort gelesen werden.

Dataflow

Der Datenfluss zwischen Front und Backend ist stark durch das Competition-Objekt geprägt. Da in der MongoDB die Competition als Ganzes in einem Dokument abgelegt wird, werden über die REST-Schnittstelle die Competitions gesamthaft aktualisiert. Dies hat sich im Verlauf des Projektes als nicht optimal erwiesen und wird in Zukunft angepasst. Durch diesen Constraint wäre es im Backend nur über tiefe Vergleiche der Datenstruktur möglich, herauszufinden, welcher Teil der Competition geändert hat und welche Rider davon betroffen sind. Um trotzdem feingranular auf Heat-Änderungen einzugehen und den betroffenen Ridern beziehungsweise den Followern der Rider eine Notification zu senden, wird diese bereits im Frontend erstellt, da dort die Information, welcher Teil der Competition geändert hat, noch vorhanden ist.

Websocket



Das proprietäre Protokoll kennt folgende Messagetypen:

- Authentication
- Notification
- Subscription
- Data

Die WebSocket-Connection wird verwendet um CompetitionEvents beim Starten, Stoppen oder Abschliessen eines Heats zu versenden. Um solche Events zu Versenden muss zuerst eine Athentifizierungs-Message an das Backend gesendet werden, um der WebSocket-Session einen User zuteilen zu können. Die Auth-Message beinhaltet lediglich das Bearertoken, welches mit dem angemeldeten User in der IndexedDB abgelegt ist. Jeder Client kann sich auf ausgewählte Rider subskribieren, diese Subscription wird per Subscription-Message an das Backend gesendet. Empfängt das Backend eine Notification-Message (Action: start | stop | save) wird diese Notification-Message aufbereitet und an alle Sessions gesendet, die eine Subscription auf einen betroffenen Rider haben. Ausserdem generiert das Backend aus diesen Notification-Messages die Rider History Items, legt diese in die DB ab und versendet sie an alle Clients als Data-Message. Wird eine Competition über die REST-Schnittstelle aktualisiert, wird die entsprechende Competition auch mittels Data-Message an alle Clients gesendet.

- Subscription-Message

```
{
  "messageType": "subscription",
  "payload": {
    "riderIds": [
      "615b4db8be111df6024475d6",
      "615b52ddbe111df6024475e3",
      "615b51d6be111df6024475e0",
      "615b4f5dbe111df6024475db",
      "615b4fb2be111df6024475dc",
      "615b5188be111df6024475df",
      "615b4ffdbe111df6024475dd",
      "615b507bbe111df6024475de",
      "6132710cfc13ae15b3000002",
      "615b5221be111df6024475e1",
      "615b4f0fbe111df6024475d9",
      "615b4caabe111df6024475d4",
      "615b5281be111df6024475e2",
      "613366d3fc13ae5ce800000a"
    ]
  }
}
```

- Authentication-Message

```
{
  "messageType": "authentication",
  "payload": {
    "bearerToken":
      "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJmcml0ejMyIiwiZXhwIjoxNjMzMjQyMDk5LCJpYXQiOjE2MzM2
      MjQwOTl9.Ah0JSeA4T-
      AliF6ULy7huq4x3QrPIc010wi6nD9c9IbXKevp_uxvMcMjrQcMF2qogK5CFCSQpao4yhULFpt63g"
  }
}
```

- Notification-Message

```
{
  "messageType": "notification",
  "payload": {
    "surfEvent": "6147a982489d233743178f52",
    "surfEventName": "oakwood jam 2021",
    "division": "kid",
    "round": 0,
    "heat": 0,
    "topic": "heat",
    "action": "start",
    "riders": [
      "613366d3fc13ae5ce800000e",
      "613366d3fc13ae5ce8000009",
      "613366d3fc13ae5ce8000008",
      "613366d3fc13ae5ce800000d"
    ],
    "results": [],
    "timestamp": "1633624158433",
    "link": "/event/oakwood-jam-2021-6147a982489d233743178f52/competition/kid"
  }
}
```

IndexedDB

Die IndexedDB wurde zur Datenhaltung für das Offline-State-Management eingeführt. Mit Hilfe des Wrappers [Dexie.js](#) werden Userdaten, Riderdaten und Competitions abgelegt. Mit dem Einführen der PWA-Funktionalität werden unsere Applikationsdaten nun gecached und müssten nicht mehr zwingend in der IndexedDB abgelegt werden. In einem nächsten Schritt könnte die IndexedDB ausgebaut und die im Offline-Modus zu postenden REST-Requests und Websocket-Messages lediglich im Localstorage ablegen werden.

Spring Boot Server

Die Spring Boot Applikation wurde möglichst schlank gehalten. Sie ermöglicht lediglich das Persistieren der Applikationsdaten in der MongoDB und verteilt Websocket Notification- sowie Data-Messages an die jeweiligen Abonnenten. Ausserdem generiert die Spring Boot Applikation "eventTimelines" und timelinelitem, aus den erhaltenen Websocket Notification-Messages, persistiert diese in der MongoDB und stellt sie über REST und Websocket Data-Messages zur Verfügung.

MongoDB

Für die Persienz kommt ein MongoDB zum Einsatz. Wir haben die Applikationsdaten sehr grobganular auf fünf Collections aufgeteilt.

- riderEntity
- competition

- eventTimeline
- surfEvent
- user

3rdParty API

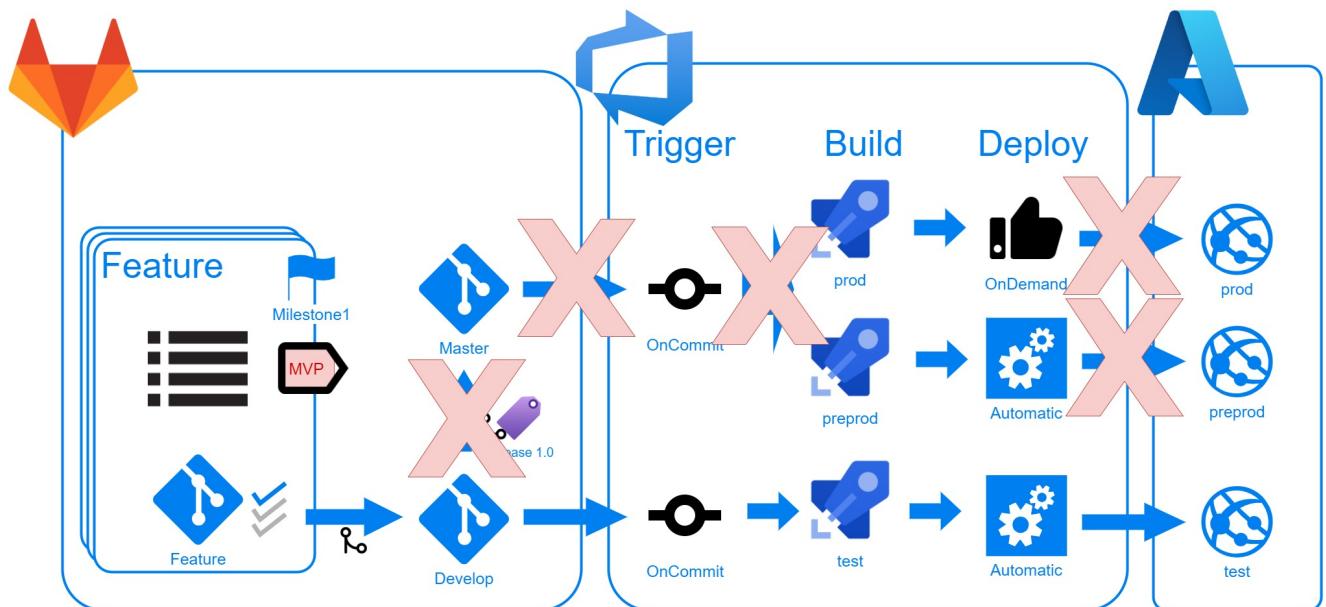
Aare.guru

Um Wetterdaten zu einem Ort eines Surf-Events zu erhalten, wurde die API von [aare.guru](#) verwendet. Die [Schnittstelle](#) wird über einen HTTP Request angesprochen und enthält die folgenden Daten:

- Name der Messstation
- Zeitstempel
- Flusstemperatur
- Lufttemperatur
- Fließgeschwindigkeit
- Wettervorhersage

Für die Anzeige der aktuellen Wetterkonditionen benutzen wir die Open Source SVG-Animationen von [Metecons](#).

Deployment



Die geplante Deployment Pipeline für die Testumgebung wurde sehr früh während der Projektdurchführung umgesetzt, damit Änderungen schnellst möglich auf verschiedenen Endgeräten getestet werden können. Die weiteren Umgebungen wurden bisher nicht umgesetzt. Da das Ziel, eine erste App-Version bereits im September am "RiversurfJam Thun 2021" einzusetzen, nicht erreicht wurde, war eine weitere Umgebung bisher nicht nötig.

Weiter zu: Eingesetzte Technologien

Eingesetzte Technologien

Für die Realisation des Projektes wurde ein separates Backend- und Frontend-Projekt erstellt. Die Studierenden setzten im Backend Spring Boot ein, welches mit Hilfe von MongoDB Daten persistiert. Aufgrund des Frontend-Fokus des Projekts wird nicht näher auf das Backend eingegangen. Im Frontend entschieden sich die Studierenden für den Einsatz von Angular. Dies, weil es einigen Studierenden der Gruppe am wichtigsten war, das Framework für den künftigen beruflichen Einsatz genauer kennenzulernen und da insgesamt das meiste Know-How innerhalb der Gruppe bei Angular vorhanden war.

Inhalt

[[TOC]]

Zusammenarbeit

Die Gruppe einigte sich früh auf den Einsatz von Gitlab, welches eine Art agile storybasierte Zusammenarbeit ermöglichte. Zunächst wurden gemeinsam Stories definiert und diese anhand der Spezifikation in die Kategorien MVP, Nice To Have, etc. unterteilt. Studierenden lag es anschliessend frei, sich selbst einer Story anzunehmen oder diese im Pair Programming abzuarbeiten. Oft wurde in der Gruppe individuell, als auch im Pair Programming (zu zweit, dritt oder gar zu viert) zusammengearbeitet. Über Azure wurde eine Build Pipeline aufgebaut, welche bei jedem Merge (nach einem erfolgreichen Pull-Request) auf den Develop-Branch, getriggert wurde.

Angular

Der Wahl im Frontend fiel wie bereits erwähnt schnell auf Angular. Das Framework wird bei vielen Arbeitgebern der Studierenden eingesetzt und war somit ein klarer Favorit. Die Studierenden Alain Messerli und Igor Stojanovic konnten vor dem CAS keine Erfahrung in Angular ausweisen. Die Erfahrung in der Gruppe brachten Raphael Gerber und Marius Stalder. Der Einsatz von React und Vue wären auch geeignete Hilfsmittel für die Erreichung der Ziele gewesen. Wir empfanden jedoch den "opinionated" Ansatz von Angular als angenehm, da so klare Strukturen vom Framework vorgegeben werden. Wir bedienten uns bei der Erstellung des Projektes an der Komponenten-Library [Angular Material](#), welche als Basis verwendet wurde und anschliessend den eigenen Bedürfnissen nach angereichert wurde.

SCSS

Die Projektgruppe legte bei der Erstellung des Projektes viel Wert auf das visuelle Erscheinungsbildes der Applikation. So wurde die Aufbau-Logik und das Erscheinungsbild der Komponenten mit sehr viel händisch geschriebenem SCSS-Code versehen.

Eigene Karussell-Komponente

Leider waren wir nicht in der Lage eine geeignete Karussell-Library in Angular zu finden, was Marius Stalder dazu verleiten liess, eine eigene Komponente zu schreiben, welche auf mobilen Endgeräten auch Swipe-Bewegungen registrieren und verarbeiten kann. Diese basiert auf Hammer.js und ermöglicht es, eine Liste von Child-Komponenten horizontal mit Swipe-Bewegungen durchzublättern.

Web APIs

Vibration API

Die [Vibration API](#) wird bei eingehenden Benachrichtigungen verwendet, um dem Benutzer auf der mobilen Ansicht ein haptisches Feedback zu geben. Die API wird im **UserNotificationService** verwendet.

Notification API

Die [Notification API](#) wird ebenfalls dazu verwendet, die eingehenden Benachrichtigungen als Benachrichtigung aus dem Browser heraus darzustellen.

Die API wird ebenfalls im **UserNotificationService** verwendet. Speziell an dieser API ist, dass zunächst die Erlaubnis vom Benutzer eingeholt werden muss, um Benachrichtigungen darzustellen:

```
constructor() {
    this.hasVibrationSupport = !!window.navigator.vibrate;
    this.hasNotificationSupport = !!window.Notification;
    if (this.hasNotificationSupport) {
        this.hasGrantedNotificationSupport = Notification.permission ===
        "granted";
        if (!this.hasGrantedNotificationSupport) {
            Notification.requestPermission().then(() => {
                this.hasGrantedNotificationSupport = Notification.permission ===
                "granted";
            });
        }
    }
    this.notificationData.pipe(
        tap(val => {
            if (this.hasNotificationSupport && this.hasGrantedNotificationSupport)
            {
                try {
                    const notification = new Notification('RiverSurf', {
                        body: val.content,
                        tag: val.surfEventName,
                        icon: '/assets/icons/icon-256x256.png'
                    });
                } catch {}
            }
            if (this.hasVibrationSupport) {
                window.navigator.vibrate(400);
            }
        })
    ).subscribe(val => {
```

```

        this.notifiedNotificationData.next(val);
    });
}

```

Drag & Drop

Die [Drag & Drop API](#) wurde über die [Angular Material CDK \(Component Dev Kit\) Drag & Drop](#) realisiert.

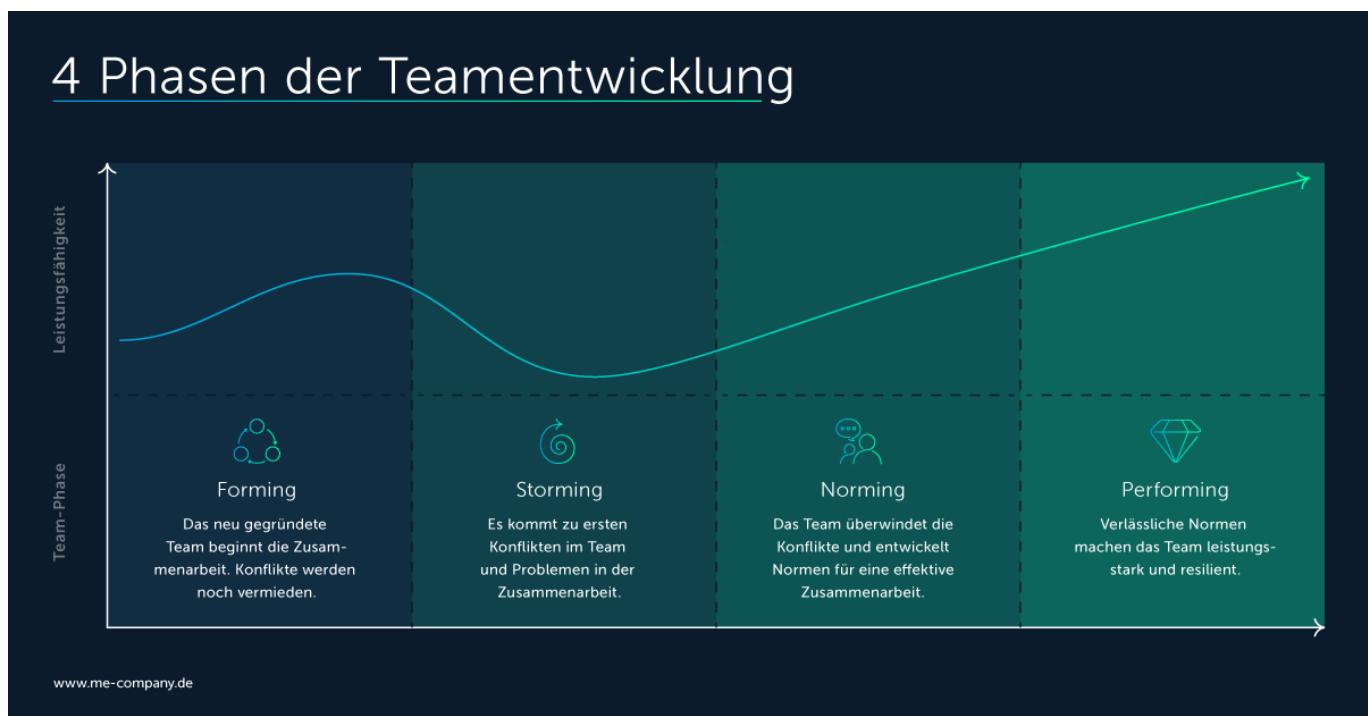
Dabei können in der Wettbewerbsansicht die Rider per Drag & Drop in die Heats zugeteilt werden ([siehe Verwaltung der Resultate als Jury](Scope des Projektes/Funktionen als Jury#will-resultate-verwalten)).

Herausforderungen

Wir haben das Projekt RiverSurf als sehr anspruchsvolles und zeitintensives Projekt wahrgenommen, woran wir jedoch stets mit grosser Freude und Motivation gearbeitet haben. Das zunächst einfache klingende Projekt einer Resultateverwaltung wurde mit vielen Features angereichert, welche User begeistern können und zum Verweilen einladen.

[[TOC]]

Herausforderungen als Team



Norming

Da sich die Gruppe bereits von einem früheren CAS kannte, war eine Zusammenarbeit naheliegend. Zwar hatten wir bis zu RiverSurf an keinem Projekt gemeinsam gearbeitet, jedoch konnte man sich ein wenig (wenn auch primär remote). Ideen waren in der Gruppe viele vorhanden und somit mussten wir uns auf ein

gemeinsamen Projekt einigen. Dies führte oft zu Diskussionen. Da wir jedoch mit RiverSurf eine echtes Bedürfnis erkannten, fiel die Wahl schlussendlich darauf.

Storming

Mit der Wahl des Projektes wurde jedoch nicht unbedingt alles einfacher. Schon früh merkten wir, dass die Vorstellungen innerhalb des Projektes weit auseinander gehen. Dies führte zu einem fast lärmenden Zustand, da sehr viele Aspekte der Applikation oberflächlich diskutiert wurden und die Anzahl der Ideen täglich wuchs.

Forming

Nach den Sommerferien starteten wir mit der Umsetzung der Applikation. Von hier an wurde vieles klarer. Die ersten Wochenenden wurde zu viert gleichzeitig an der Applikation gebastelt. Dies war zwar ein zeitintensives Unterfangen, jedoch konnten Meinungsverschiedenheiten gleich diskutiert werden. Das Commitment der ganzen Gruppe war nach der Erstellung des "Grobkonstrukts" deutlich höher.

Performing

Die Performing-Phase wurde klar mit der Definition des Grobkonstruktes eingeleitet. Von hier an war das Commitment der Gruppe riesig. Zusätzlich mit den absolvierten Usability Tests am diesjährigen RiverSurf Jam in Thun war die Gruppe überzeugt, eine Superlösung für viel manuelle Arbeit bieten zu können. Auch war ein guter Skill-Mix innerhalb der Gruppe ideal um sich laufend zu hinterfragen und die Applikation zu verbessern.

Technische Herausforderungen

Resultat-Anzeige und -Verwaltung

Das Herzstück der Applikation bestand aus der einfachen Verwaltung und Darstellung der Resultate an einem Wettkampftag. Besucher des Events sollten in der Lage sein, sich jederzeit über einen Event (oder einen Rider) auf dem Laufenden zu halten, ohne eine Applikation installieren zu müssen. Hier haben wir bei der "Customer Journey" schon sehr früh angesetzt. User sind in der Lage mit einem QR-Code ein Resultate-Board zu scannen und werden direkt auf dieses geleitet. Mit einem Klick (auf das Herz) können sie sich gleich einem Rider, bei welchem sie im Bild bleiben möchten, folgen. Auch war es uns wichtig, einen Wettkampfverlauf für den Besucher so einfach wie möglich darzustellen. Diese wird bei grossen Screens mit Hilfe von SVG-Zeichnungen gemacht. Mit einem Klick auf einen Rider, werden die übrigen Rider aus dem Fokus genommen, und der Turnierverlauf des angewählten Riders wird fokussiert dargestellt:

RiversurfJam Thun 2021

Male Female



Das Ganze war Mobile aufgrund der beschränkten Platzverhältnisse schwierig umzusetzen. Die Gruppe entschied sich deshalb dazu, die eigens programmierte Karussell-Komponente zu verwenden. Bei einem Tap auf einen Rider sollte jedoch das Karussell an die richtige Stelle (zum richtigen Heat) springen. Dies ermöglicht eine übersichtliche Darstellung der Resultate auf einem schmalen Screen:

RiversurfJam Thun 2021

Male

Female

SEEDING ROUND

Heat 3

Agace M.

99

Lesley N.

30

Merna B.

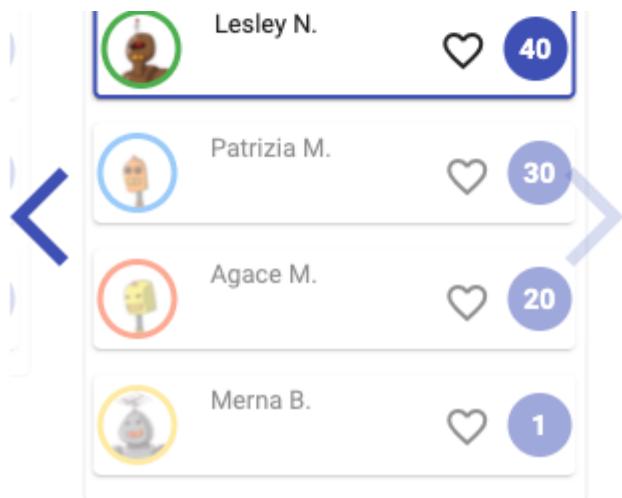
30

Patrizia M.

30

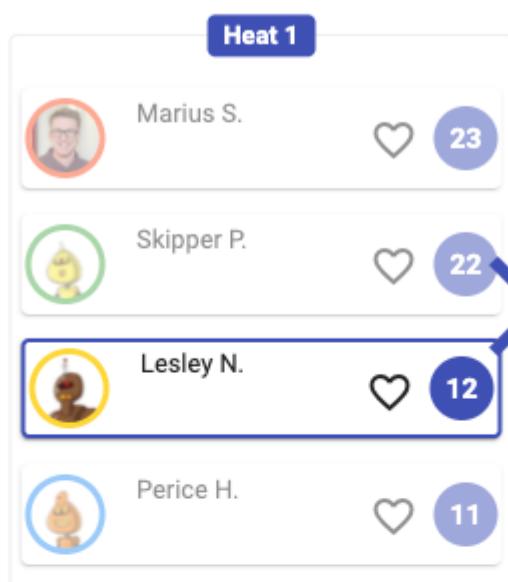
ROUND 1

Heat 4



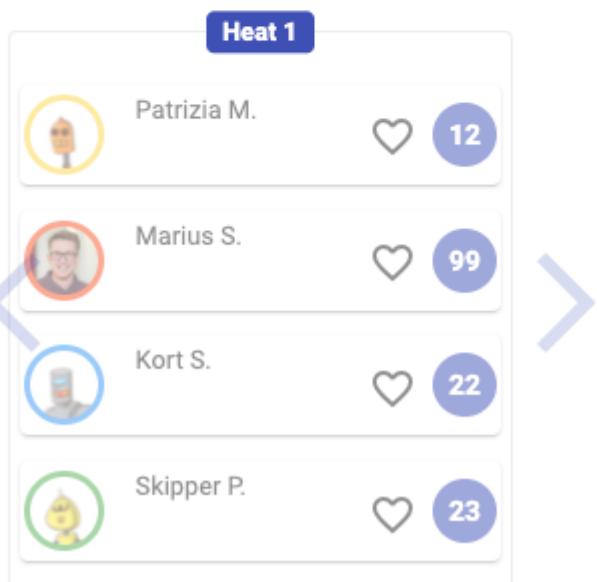
SEMIFINALS

Heat 1



FINALS

Heat 1



Zunächst war unser Gedanke die Verwaltungssicht des Turnieres analog der Desktop-Ansicht der Resultate zu erstellen. Jedoch wurde im Usability Test schnell klar, dass dieser Weg alles andere als einfach für den Anwender ist. Das Scrollen und Einteilen der Riders in einem Heat war zu umständlich und der Informationsgehalt auf der entsprechenden Seite zu gross. Somit entschied sich die Gruppe für eine Unterteilung und Darstellung einer Runde anstatt den ganzen Wettbewerb:

| Heat | Rider Name | Avatar | Score |
|--------|----------------------|--------|-------|
| Heat 1 | Creighton Greensides | | |
| Heat 1 | Gerick Artharg | | |
| Heat 1 | Normand Lardiner | | |
| Heat 1 | Roanne Moxom | | |
| Heat 2 | Clarie O'Nion | | 1 |
| Heat 2 | Luciano Dy | | 1 |
| Heat 3 | Dun Satterly | | 20 |
| Heat 3 | Doralin Ormestone | | 30 |
| Heat 3 | Gillie Salzberg | | 25 |
| Heat 3 | Raimund Tesdale | | 10 |

Die Features werden genauer im Kapitel Die kleinen feinen Features beschrieben.

WebSocket

Eine der grösseren Herausforderungen war die Authentifizierung einer WebSocket-Connection. Da im Frontend Notification-Messages wie z.B. "Heat hat gerade gestartet" oder "Heat wurde beendet" erzeugt werden können, muss sichergestellt werden, dass Notification-Messages nur von authentifizierten Benutzern akzeptiert werden. Der erste Ansatz, dass sich das Frontend, sobald der User ein Login macht, über REST eine authentifizierte WebSocket-Session reserviert und vom Backend das entsprechende Session-Token erhält um eine neue Connection aufzubauen, ist gescheitert. Das Wechseln der Session hat nicht sehr stabil funktioniert. Der zweite Ansatz, dass über die bestehende Connection das JWT-Bearertoken als Authentication-Message gesendet wird und der Session so ein User zugeordnet werden kann funktioniert nun sehr stabil und hat sich als wesentlich elegantere Lösung bewährt.

Usability Test

Aus Termingründen wurden die beiden Testszenarien vorgängig aufgezeichnet. Die Durchführung hat während des RiversurfJam Thun mit einem Mitglied der Jury stattgefunden. Die Rahmenbedingungen waren somit sehr realistisch und der User wurde teilweise durch Fragen oder vorbeigehende Passanten abgelenkt.

Szenario 1

- Anspruchsgruppe: Eventbesucher
- Device: Mobiltelefon (Galaxy S9)

- Video: [Video zu Testcase 1](#)

Du bist ein Fan vom Rider „Pawlett Nahum“. Er ist ein Riversurfer und hat sich auf <https://test.riversurf.app> als Rider registriert

- a) Finde heraus wann Pawlett geboren ist
- b) Sorge dafür, dass du immer auf dem Laufenden bist, wenn sich bei Pawlett etwas tut.

Szenario 2

- Anspruchsgruppe: Jurymitglied
- Device: Laptop (Dell-xps 13“)
- Video: [Video zu Testcase 2](#)

Du bist Mitglied der Jury vom RiverSurf-Event, der in Thun im Jahr 2021 stattfindet.

- a) Logge dich auf <https://test.riversurf.app> mit

- E-Mail: jury@riversurf.app
- Passwort: jury1234

ein.

- b) Teile die Riders der Division „male“ in der ersten Runde in die Heats ein.

Weiter zu: Lessons Learned

Lessons Learned

Wir können klar ein gemeinsames Fazit ziehen: "**If you want to go fast, go alone. If you want to go far, go together**".

Das Projekt RiverSurf war insgesamt eines der spannendsten Projekte, welches wir in die Realität umsetzen konnten. Nicht immer war die Einigung auf Features und deren Implementierung einfach, jedoch schuf der Raum des kritischen Diskurses die Möglichkeit, uns ständig als Gruppe weiterzuentwickeln. Wir hatten grosses Glück, in der Gruppe in jedem Teilbereich einen starken Vertreter zu haben, der gewillt war, sein Wissen weiterzugeben und somit die Gruppe voranzubringen. Auch hat uns die Umsetzung vieler Features (z.B. SVG-Zeichnung der Resultate, WebSocket Benachrichtigungen, Wettkampf-Verwaltung, etc.) viel Freude bereitet und zu einem grossen Lerneffekt geführt. Des Weiteren konnten wir im Rahmen des Projektes viele im CAS kennengelernte Konzepte im Frontend (Immutable State, ES6-Sprachkonstrukte, Angular Modularisierung, Services, etc.) in die Praxis umsetzen.

Was wir das nächste Mal besser machen würden

Vom gleichen zu reden, heisst nicht immer das gleiche zu verstehen. Uns hat die Ideation-Phase im Usability-Kurs klar aufgezeigt, dass wir häufig vom gleichen sprachen, jedoch nicht das gleiche darunter verstanden. Hätte es das Zeitbudget erlaubt, hätten wir die ganze Applikation graphisch geprototyped, getestet und anschliessend implementiert. In der Realität waren wir jeweils ziemlich schnell in der Umsetzung, da uns für die anderen Aspekte die Zeit fehlte und das Umsetzen grosse Freude bereitete.

Glossar

| Begriff | Erklärung |
|-------------|--|
| Competition | Ein Wettkampf. Je nach Event sind verschiedene Wettkämpfe für Frauen, Kinder oder Männer möglich |
| Jam, Event | Veranstaltung |
| Heat | Eine Wettkampfrunde ist in mehrere sog. Heats eingeteilt, wo eine Gruppe von Riders gegeneinander antritt. In der Regel kommen die beiden Besten der Gruppe weiter in die nächste Runde. |
| Rider | Teilnehmer einer Veranstaltung |
| River Surf | Flusssurfen |
| Round | eine Wettkampfrunde |