

SwimFit

Ein Trainingsanalyse Tool für Schwimmer

Meilenstein 4

EIS

Entwicklung Interaktiver Systeme

ausgearbeitet
von

Franziska Schmidt

Inhaltsverzeichnis

1.0 Datenstruktur	S. 3
1.1 Collection: Schwimmer	S. 3
2.0 WBA Modellierung	S. 3
2.1 REST API Spezifikation	S. 3
2.2 Server	S. 4
2.3 Client	S. 6
3.0 Anwendungslogik	S. 6
3.1 Trainingsplan	S. 7
3.2 Berechnungen der Pulswerte	S. 7
4.0 Design, Testen und Entwickeln	S. 8
4.1 Work Reengineering	S. 8
4.2 Screen Design Standards	S. 9
4.3 Prototypen UI	S.10

1.0 Datenstruktur

Es wurde sich für das Datenformat Javascript Object Notation (JSON) entschieden. Die Daten werden in einer MongoDB Datenbank abgespeichert. "Die Grundlagen der Speicherung von Informationen in dieser Datenbank basiert auf Dokumenten, die im BSON- Format vorliegen"¹. Die Datenbank wird nochmal untergliedert in Collections, in denen einzelne Dokumente hinterlegt sind. Im Folgenden werden die verschiedenen Collections mit einem Beispieldokument dargestellt.

1.1 Collection: Schwimmer

```
{
  "_id" : ObjectId("4dcd3ebc9278000000005158"),
  "date" : ISODate("2011-05-13T14:22:46.777Z"),
  "name" : "Sophie Müller",
  "geburtstag" : "26.06.1991",
  "Kenntnisse" : "Kurs 1",
  "Verein" : "Campussport",
  "Ruhepuls" : "90",
  "Belastungspuls" : "175"
}
```

2.0 WBA Modellierung

2.1 REST API Spezifikation

Alle Ressourcen der REST API werden mit dem anwendbaren HTTP Methoden wie folgt beschrieben:

/schwimmer

GET: Die Schwimmer-Ressource mit der eindeutigen ID {schwimmer} wird im Response gesendet.

POST: Eine neue Schwimmer-Ressource wird unter eindeutiger, automatisch generierter ID in der Datenbank gespeichert.

¹ Stephan Mattescheck, Erik Lipperts : Node.js S.375

PUT: Die Schwimmer-Ressource mit eindeutigen ID {schwimmer} wird die im Request enthaltenen Informationen in der Datenbank aktualisiert.

DELETE: Die Schwimmer-Ressource wird aus der Datenbank gelöscht.

/trainer

GET: Die Trainer-Ressource mit der eindeutigen ID {trainer} wird im Response gesendet.

POST: Eine neue Trainer-Ressource wird unter eindeutiger, automatisch generierter ID in der Datenbank gespeichert.

PUT: Die Trainer-Ressource mit eindeutigen ID {trainer} wird die im Request enthaltenen Informationen in der Datenbank aktualisiert.

DELETE: Die Trainer-Ressource wird aus der Datenbank gelöscht.

/training

GET: Die Listenressource aller Trainings-Ressourcen wird im Response gesendet

/modul

GET: Die modul-Ressource mit der eindeutigen ID {modul} wird im Response gesendet.

POST: Eine neue modul-Ressource wird unter eindeutiger, automatisch generierter ID in der Datenbank gespeichert.

PUT: Die modul-Ressource mit eindeutigen ID {modul} wird die im Request enthaltenen Informationen in der Datenbank aktualisiert.

DELETE: Die modul-Ressource wird aus der Datenbank gelöscht.

/clubs

GET: Die Listenressource aller Club-Ressourcen wird im Response gesendet

/club

GET: Die club-Ressource mit der eindeutigen ID {modul} wird im Response gesendet.

POST: Eine neue club-Ressource wird unter eindeutiger, automatisch generierter ID in der Datenbank gespeichert.

PUT: Die club-Ressource mit eindeutigen ID {modul} wird die im Request enthaltenen Informationen in der Datenbank aktualisiert.

DELETE: Die club-Ressource wird aus der Datenbank gelöscht.

/competition

GET: Die Listenressource aller competition-Ressourcen wird im Response gesendet

/equipment

GET: Die Listenressource aller equipment-Ressourcen wird im Response gesendet

2.2 Server

Der Server beinhaltet: Node.js Server mit Express und Jade als Middleware und eine MongoDB Datenbank.

Für die Umsetzung des Servers wurde sich für einen Node.js Server entschieden, weil diese in einfachster Form aus WBA2 bekannt ist. Des Weiteren wird dieses als Back-End empfohlen aufgrund:

- Eingebaute Server Funktionen
- Gutes Projektmanagement
- Eine schnelle JavaScript engine (V8)
- Asynchrones event gesteuertes Programmiermodell²

Durch Cocoa's NSJSONSerialization können die im JSON Format gespeicherten Daten im Client der auf IOS basiert verwendet werden.

Zum Server gehören folgende Dateien:

collectionDriver.js

fileDriver.js

index.js

package.json

Ordner: public mit index.html

collectionDriver.js kümmert sich um die MongoDB Funktionen. fileDriver.js kümmert sich um die HTTP Methoden. Index.js beinhaltet den kompletten Aufbau des Back-Ends inklusive des Server aufbaus, HTTP Methoden und Datenbankerstellung.

```
var http = require('http'),
    express = require('express'),
    path = require('path');
MongoClient = require('mongodb').MongoClient,
    Server = require('mongodb').Server,
    CollectionDriver = require('./collectionDriver').CollectionDriver;
FileDriver = require('./fileDriver').FileDriver;
```

```
var app = express();
app.set('port', process.env.PORT || 3000);
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
app.use(express.bodyParser());
```

² <http://www.raywenderlich.com/61078/write-simple-node-jsmongodb-web-service-ios-app>

```

var mongoHost = 'localhost';
var mongoPort = 27017;
var collectionDriver;
var fileDriver;

```

2.3 Client

Die Klasse Trainingsplanmodul beinhaltet die Trainingsmoduldaten, weiß wie JSON kompatible NSDictionary Objekte serialisiert und deserialisiert werden und hat 0 oder mehr Kategorien, wie in Categories.m definiert.

Die Klasse Trainingsmodule repräsentiert die Sammlung von Trainingsmodul Objekten, die vom Server geladen werden. Es wird mit dem Server kommuniziert, indem Objekte geladen und gespeichert werden.

Die Klasse Categories enthält die Liste von Kategorien, zu der ein Trainingsmodul gehören kann: Anfänger, Fortgeschrittener und Profi. Sie filtert des Weiteren die Trainingsmodule nach Kategorien.

Das Importieren wird durch folgenden Code realisiert:

```

- (void)import
{
    NSURL* url = [NSURL URLWithString:[kBaseURL
stringByAppendingPathComponent:kLocations]];

    NSMutableURLRequest* request = [NSMutableURLRequest requestWithURL:url];
    request.HTTPMethod = @"GET";
    [request addValue:@"application/json" forHTTPHeaderField:@"Accept"];

    NSURLSessionConfiguration* config = [NSURLSessionConfiguration
defaultSessionConfiguration];
    NSURLSession* session = [NSURLSession sessionWithConfiguration:config];

    NSURLSessionDataTask* dataTask = [session dataTaskWithRequest:request
completionHandler:^(NSData*data, NSURLResponse*response, NSError*error) {
        if (error == nil) {
            NSArray* responseArray = [NSJSONSerialization JSONObjectWithData:data options:0
error:NULL];
            [self parseAndAddLocations:responseArray toArray:self.objects];
        }
    }];

    [dataTask resume];
}

```

3.0 Anwendungslogik

Die Anwendungslogik muss in Verbindung mit der Domänenrecherche entwickelt werden, da es kein allgemeines Schema für die Trainingsplanentwicklung und die Pulswertbestimmung gibt. Die organisatorischen und mathematischen Grundlagen wurden durch Befragungen von Domänenexperten entwickelt.

3.1 Trainingsplan

Anzahl	Strecke (m)	Lage	Puls	Summe	Gesamt	Zeit
Einschwimmen						
1	300	bel	110	300	300	06:00
3	150	50 B-Arme / K-Beine+ 50 R- Altdeutsch+ 50 ganze Lage	120	450	750	03:30
4	25	Brust Technik	140	100	850	00:40
Hauptteil						
18	50	6xMini-Lagen, 1 min Pause, 6xHL, 1min Pause, 6x NL	140	900	1750	01:00
1	400	HL - Technik	135	400	2150	10:00
5	100	HL 25schnell+75locker, 50schnell+50locker, 75schnell+25locker, 100locker, 100schnell	140	500	2650	02:00
4	50	50 Brust gleiten, so wenig Armzüge wie möglich	130	200	2850	00:00
1	400	bel. Beine	140	400	3250	10:00
1	400	bel. Arme mit Paddles und Pullboy	140	400	3650	07:00
Ausschwimmen						
1	200	locker	110	200	3850	00:00
				0	3850	00:00

(Bild 3.1 Trainingsplan)

Ein Trainingsplan (siehe Bild 3.1 Trainingsplan) besteht aus 3 Hauptmodulen:

1. Einschwimmen
2. Hauptteil
3. Ausschwimmen

Diese Hauptmodule sind unterteilt in Schwimmmodule mit den Angaben: Anzahl, Strecke (m), Lage, Puls, Summe, Gesamt, Zeit.

Für die Umsetzung der dynamischen Anpassung des Trainingsplans wurde folgender Code für die Erstellung und Eintragung von Trainingsplanmodulen in die NSDictionary festgelegt. Dadurch kann auf die Objekte und dessen Values zugegriffen werden.

```
- (instancetype) initWithDictionary:(NSDictionary*)dictionary
{
    self = [super init];
    if (self) {
        self.modulname = dictionary[@"modulname"];
        self.modulgruppe = dictionary[@"gruppe"];
        self.strecke = dictionary[@"strecke"];
    }
}
```

```

        self.lage = dictionary[@"lage"];
        self.zeit = dictionary[@"zeit"];
        self.puls = dictionary[@"puls"];
        _categories = [NSMutableArray arrayWithArray:dictionary[@"categories"]];
    }
    return self;
}

```

Für die Datenbank werden diese mit folgender Methode in JSON kompatible Objekte umgewandelt:

```

- (NSMutableDictionary*) toDictionary
{
    NSMutableDictionary* jsonable = [NSMutableDictionary dictionary];
    safeSet(jsonable, @"modulname", self.modulname);
    //...
    return jsonable;
}

```

Für die App Darstellung wird die Gesamtstrecke in dem Objektvalue „strecke“ abgespeichert. Die Aufteilung, die im Originaltrainingsplan (siehe Bild 3.1 Trainingsplan) in Anzahl und Strecke unterteilt ist, wird dort zusammengefasst, weil in „lage“ genau beschrieben wird, welche Schwimmart wie lange zu schwimmen ist. Zum Beispiel: „50 Meter Brustbeine, 100 Meter Brustarme“. Dies ergäbe eine „strecke“ von 150 Metern.

In der Appdarstellung wird die Gesamtzeit und Gesamtstrecke durch Addition der „strecke“ und „zeit“ errechnet und ausgegeben.

Die Pulswerte werden als Prozentzahlen eingegeben aber als Pulszahl ausgegeben. Der Grund wird in 3.2 Berechnungen der Pulswerte erläutert.

3.2 Berechnungen der Pulswerte

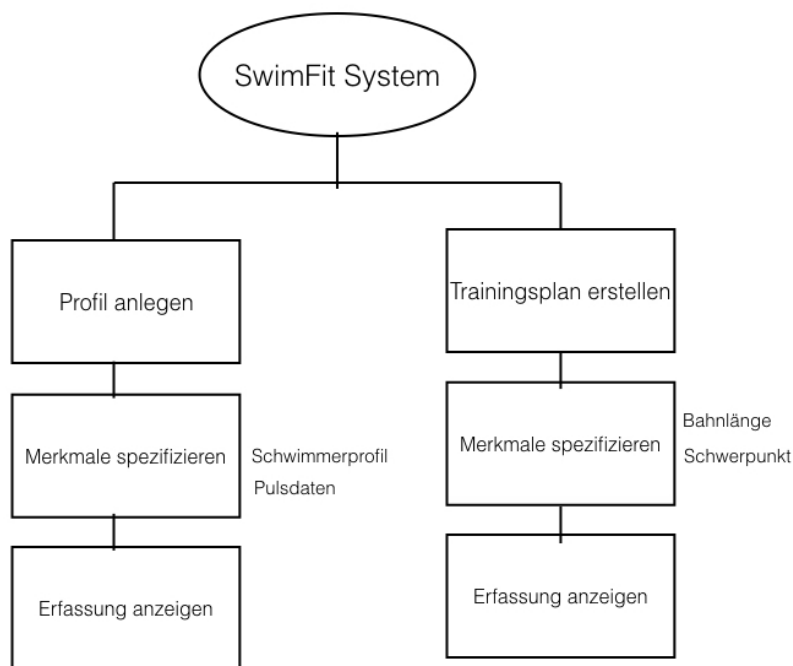
Der Pulswert im Trainingsplan wird durch den Belastungspuls und Erholungspuls ermittelt, die der Benutzer selber mit der AppleWatch bestimmt bzw. selber eingibt. Die Datenbankobjekte enthalten Prozentzahlen und dadurch kann der User individuelle Pulswertvorgaben im Trainingsplan erhalten. Der Belastungspuls ist 100 Prozent des Pulswertes also zum Beispiel 175. Der Ruhepuls stellt die Untergrenze des Pulswertes da. Dieser darf nicht unterschritten werden. Dies ist durch eine if-Abfrage gelöst.

4.0 Design, Testen und Entwickeln

Die zweite Phase des Usability Engineering Lifecycle werden die Prozesse: Design, Testing und Entwicklung durchgeführt. Diese Phase hat 3 Levels. Das erste Level „Work Reengineering“ (Mayhew 1999) wird im Folgenden behandelt.

4.1 Work Reengineering

Der Prozess „Work Reengineering“ beschreibt die Form von Neuausrichtung der momentanen Problem- bzw. Arbeitsbewältigung. Hierbei werden die zuvor durchgeführten Arbeitsschritte genutzt, um darauf aufbauend ein „Reengineered Task Organization Model“ zu entwickeln. Dieses Modell beschreibt eine Aussicht auf einen möglichen Ablauf in dem zu entwickelnden System.

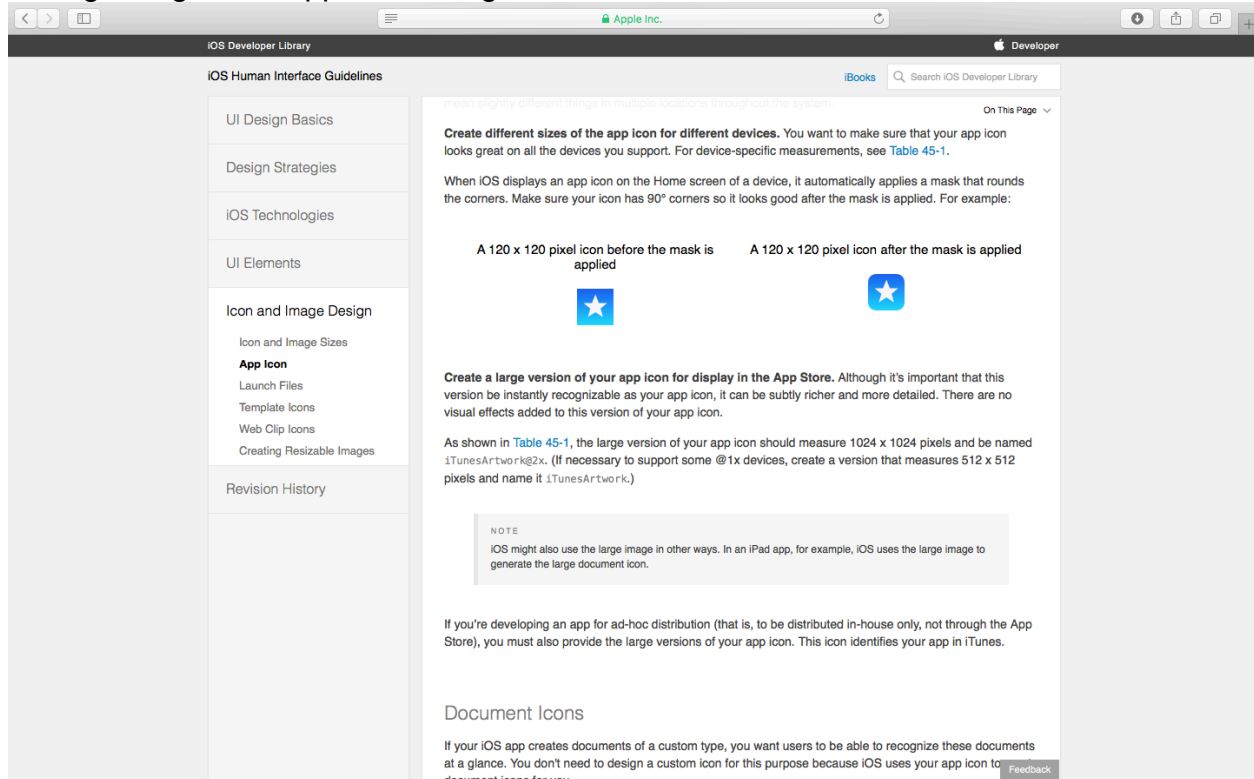


(Grafik 4.1 Reengineered Task Organization Model eines Schwimmers, der einen Trainingsplan erstellen möchte)

4.2 Screen Design Standards

Die Entwicklung eines eigenen Styleguide ist laut Mayhew für kleine Projekte nicht

zwingend notwendig. Zudem wird ein Styleguide von Apple durch die iOS Human Interface Guidelines vorgegeben³. Bild 4.2 zeigt ein Beispieldokument der iOS Human Interface Guidelines. Dort sind zum Beispiel Größe, Abstand, Farbgebung und Formgebung eines App Icons dargestellt.



(Bild 4.2 Beispiel iOS Human Interface Guidelines)

4.3 Prototypen UI

In die Phase des Designs fließen alle Faktoren ein, die zuvor durch Analysen und Modellierungen ermittelt wurden.

Im Folgenden ist ein Mockup des Prototypens zu sehen. Die Größe des Device wurde aus Übersichtszwecken vergrößert.

³ <https://developer.apple.com/> iOS Human Interface Guidelines



Anmelden

E-Mail

Passwort



Registrieren

Persönliche Daten

Name:

Geburtstag:

Geschlecht:

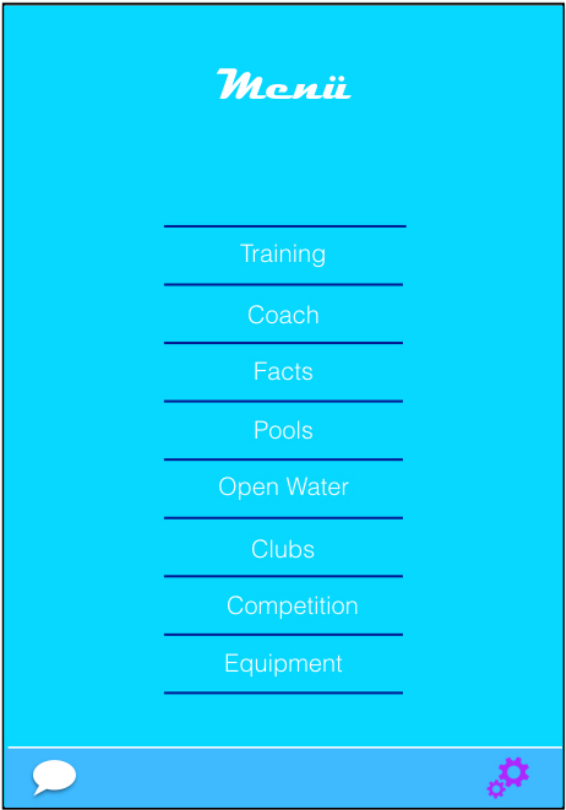
Kenntnisse:

Verein:

Ruhepuls:



erstellen



Workout

Bahnlänge:

50 m ☐

25 m ☐

Schwerpunkt:


Brust ☐

Kraul ☐

Rücken ☐

Delphin ☐

erstellen

 Menü 

Trainingsplan

Einschwimmen			
Hauptteil			
Ausschwimmen			

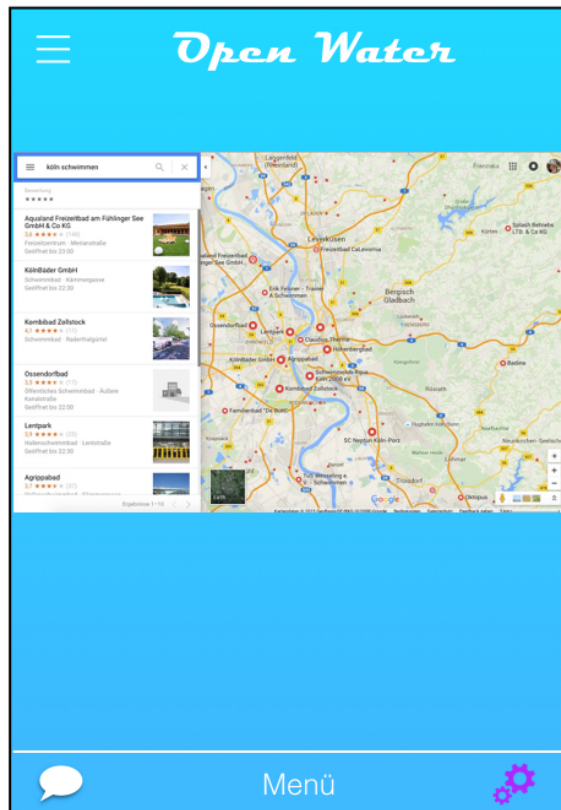
drucken

exportieren



Menü





Clubs

Uniteam

TelekomKöln

Schwimmfreunde

SV Lindenthai

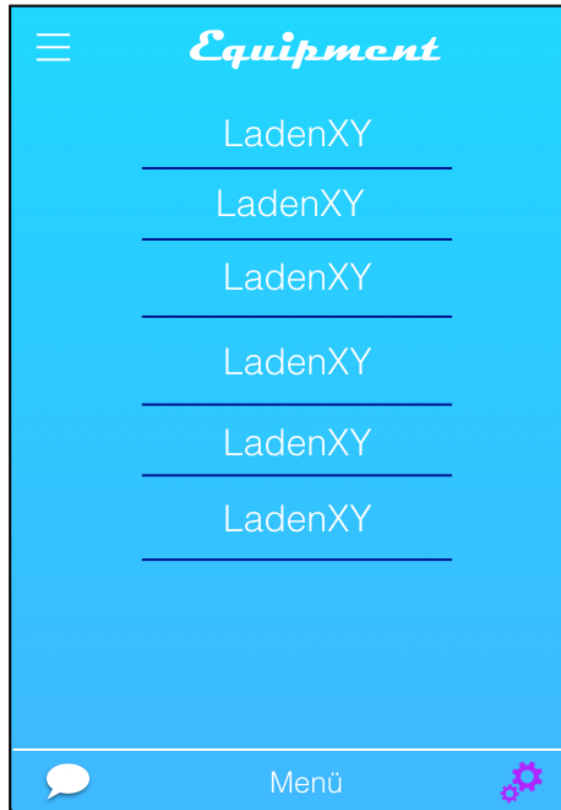
SV Zollstock

SV Spaß

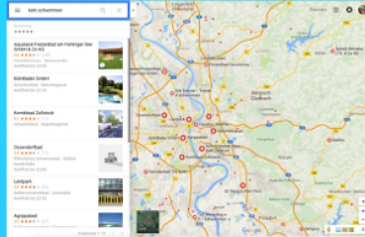


Menü





LadenXY



Beschreibung:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo

Vorteile für App Nutzer:

Lorem ipsum dolor



Menü



UniTeam



Beschreibung:

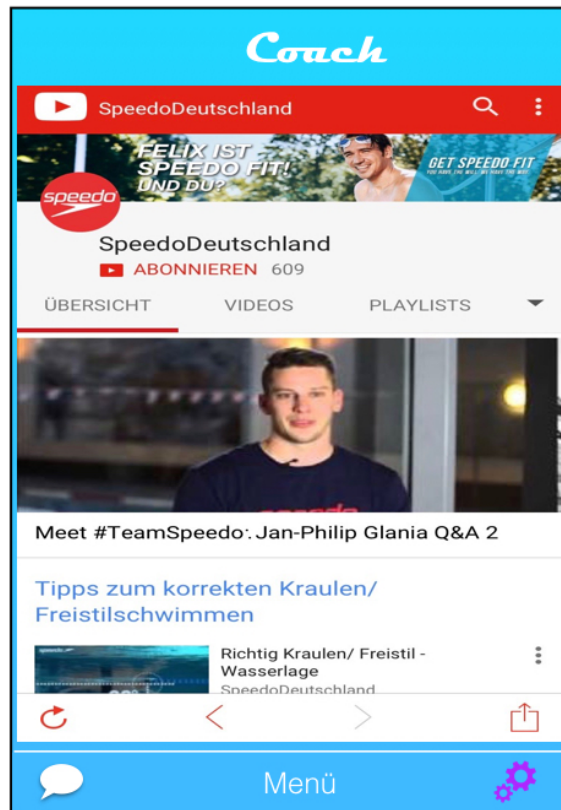
Lorem ipsum dolor sit amet,
consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut
labore et dolore magna aliquyam
erat, sed diam voluptua. At vero eos
et accusam et justo

Link:



Menü





Competition

01.12.15 10 Uhr	Name: WettbewerbXY Ort: Schwimmhalle Bochum
01.12.15 11 Uhr	Name: WettbewerbXY Ort: Berlin
03.03.16 10 Uhr	Name: toll Ort: Schwimmhalle Bochum
03.03.16 15 Uhr	Name: WettbewerbXY Ort: Köln
01.11.16 10 Uhr	Name: super Ort: Schwimmhalle Bochum

Menü

