Challenges on golf project

Time to say «Hello»

Position: Sr. Front End
Role: Dev Lead

I had to dive into golf and its rules. And I thought that I had to design a schema and front end architecture (including integration with AEM) on the whole.

I used to play golf 15+ years ago for a while in England. I enjoyed it. First time I played it on PC AT 286

Historical overview

«Gentleman Only -
Ladies Forbidden»

Golf was first mentioned in Holland - 26 Feb 1426. Men played this game using wooden clubs and feather balls. Who performed fewer hits before putting a ball into far target (small round hole) was the winner.

However it's considered that the true game was born in Scotland.

«Gowf», «Kolf» are variations of game’s name, meaning «hit».

Truly or not the game was reflected in king’s Jacob II statute in 1452. The game was forbidden because people prefered playing to war training.

There is rumor that Scottish peasants used rabbit holes for playing.

Modern golf ball (previously made of gutta-percha or feather) was manufactured in 1900. It had very good aerodynamics. So it's speed is around **270-320 km/h**.

Golf was an official olympic discipline only in 1900 and 1902. After **114** years it was added to the Olympic games programme again.

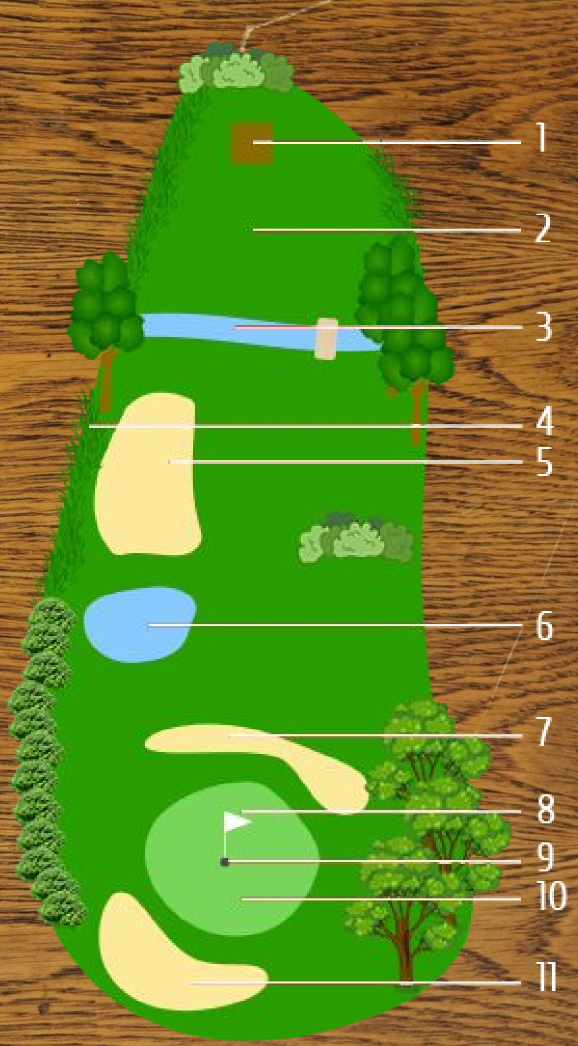
Modern game and rules are not far away from original. It looks like they have become more formal.

Game field was defined more accurately. Its complexity was changed a few times.

Basically field (course) consists of 18 holes (every hole occupies a large area containing obstacles and other difficulties for players).

To put your ball into the hole you have to carefully choose one from 7 and more clubs and make an accurate hit, taking into account wind, ground levels, grass type etc.

1	Tee (starting point, first hit)
2	Fairway (a good grass)
3 & 6	Water hazard (don't put ball into it!)
4	Rough (a bad long grass)
5 **7** **11**	Sand bunker (you should be a master to save your score)
8	Flagstick
9	The hole (you will not find a rabbit)
10	Green (much easier area, but sometimes ground levels are your enemies)



An usual golf event has around **150** participating players.

The entry fee starts from 100 $ (for simple event) and ends at **1.2k $** for season.

Almost every golfer is associated with a club. Having salary (approximately from 50k to 100k $) he/she can win **1 million $** and more on major championship. Many golfers are advertising special equipment and even products. Also there are many activities and campaigns around the world.

Thus we see that golfers are fairly wealthy people though they are not the richest.

Modern competition

- ➔ Has many stats related to the whole competition and specific players (e.g. condor, albatross, eagle, birdie, par, bogey (double/triple), driving distance, putts etc.)
- ➔ All stats are recorded in real time on the field. Copters, tracking devices (GPS for ball and player position) and other stuff are here to serve your needs.
- ➔ Match can be played even on different courses and in random order.
- ➔ Due to the nature of competition there are many places for typos.

Modern competition

- Players can play side-by-side on the single hole. Even rain or strong wind can not stop them.
- Usually it takes a few days.

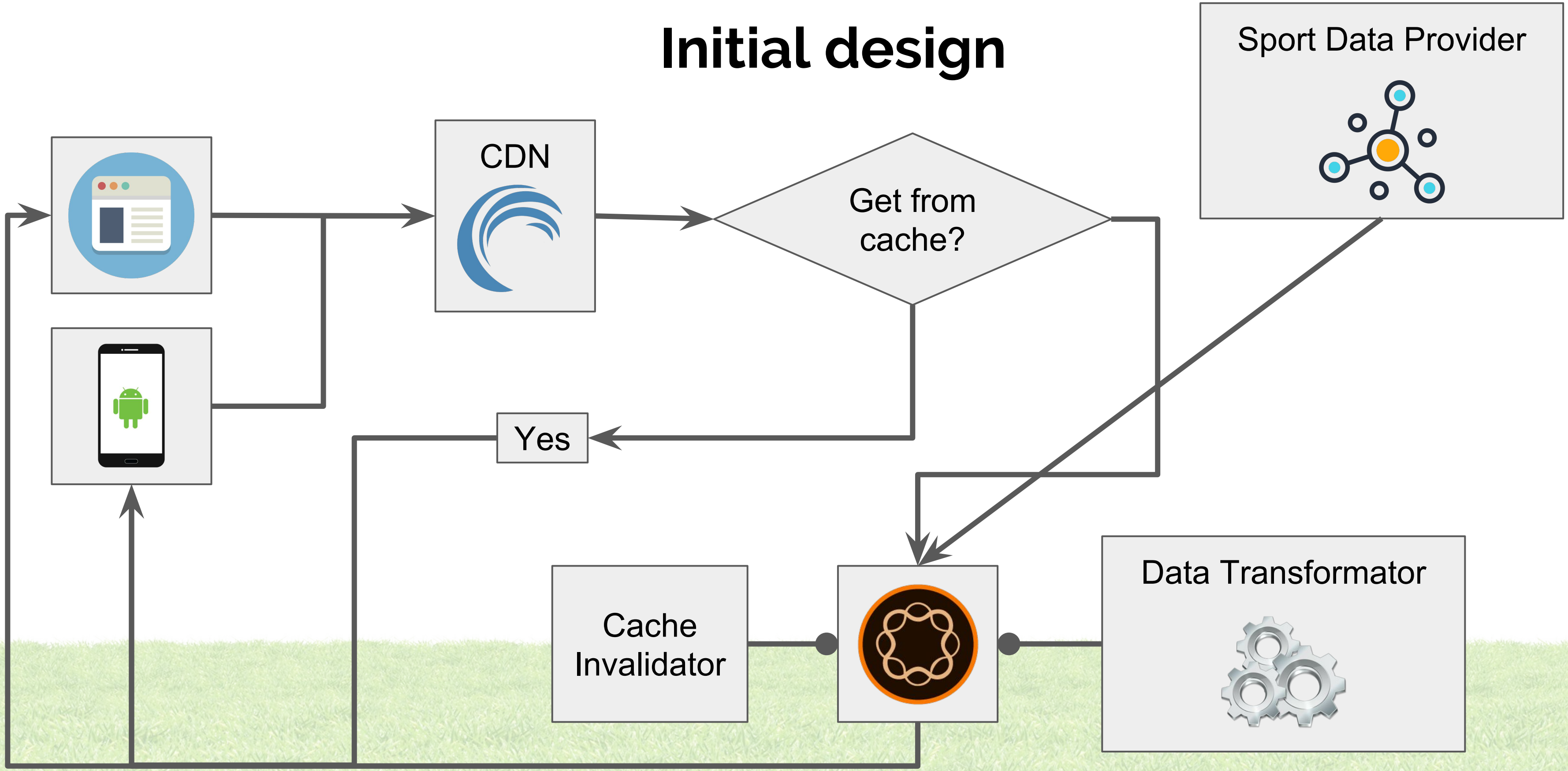
Beginning

Non-trivial golf project goals

- ➔ Be PM/DM friendly.
- ➔ Be fast. Milliseconds matter. Have KPI as low as possible.
- ➔ Have isolated and hot replaceable components.
- ➔ Live under high pressure. **1'000k** requests per day and more.
- ➔ Backup previous data.
- ➔ Be able to edit sport data quickly.

Design

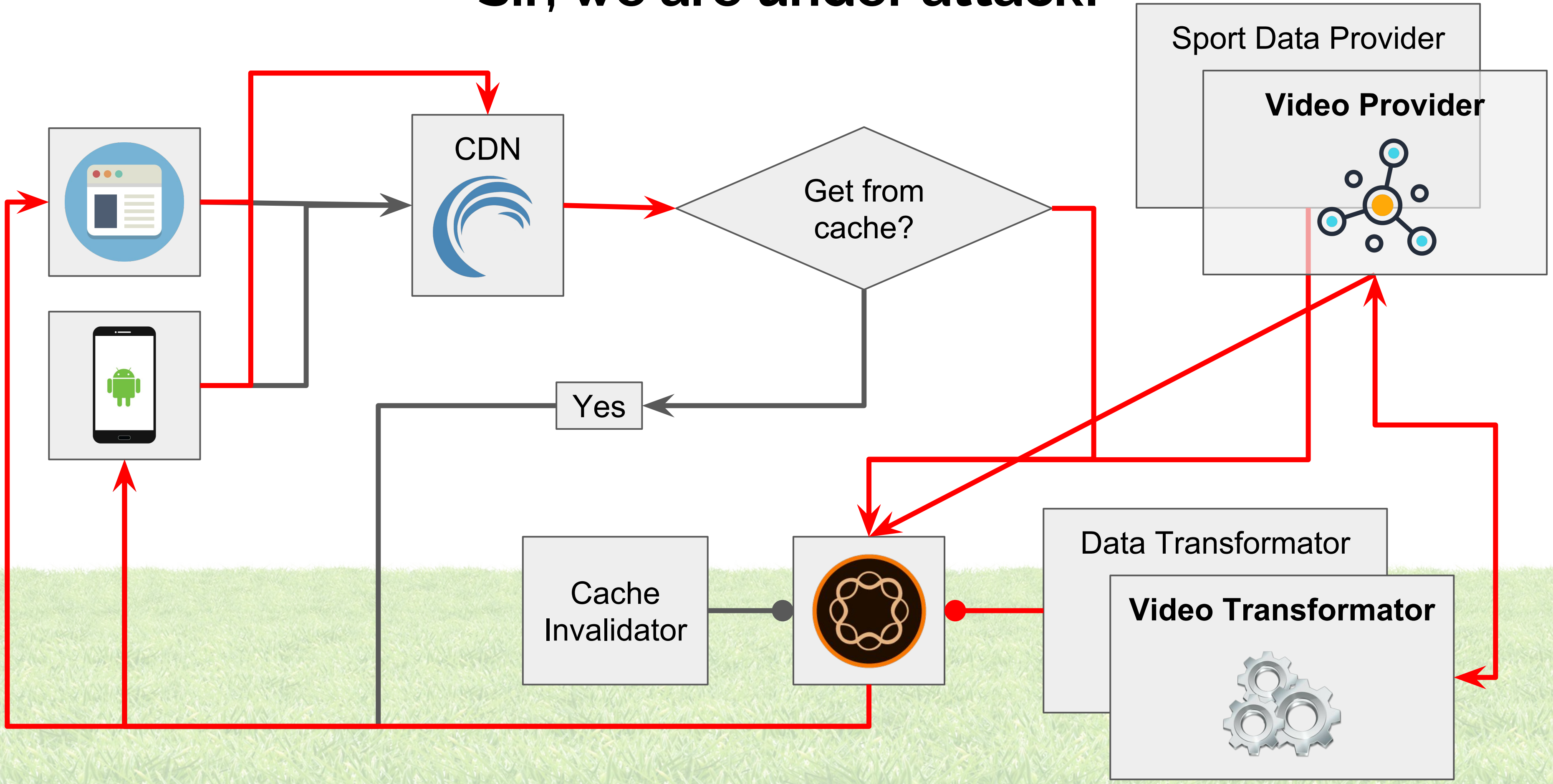
Initial design

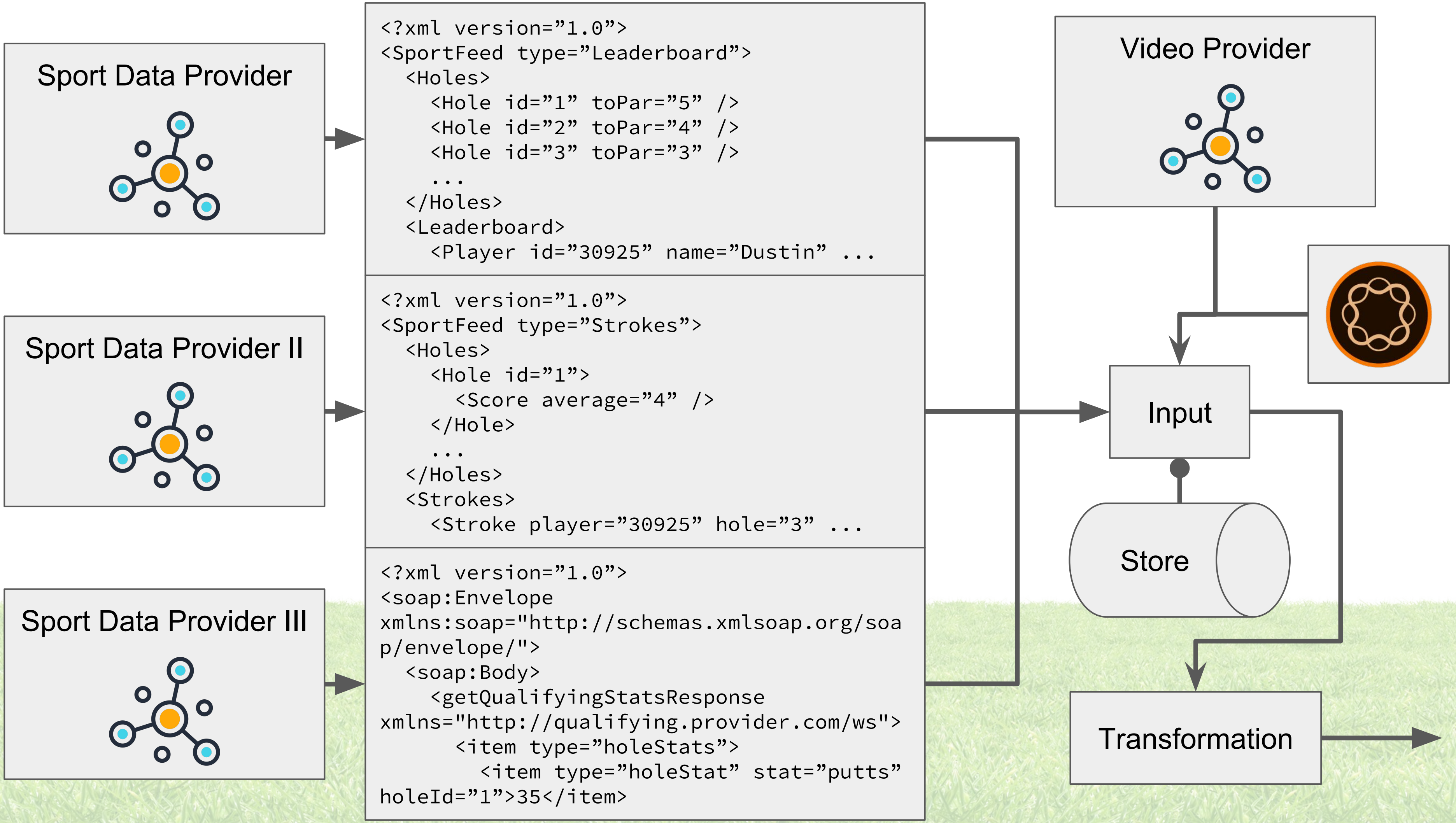


Features

- ➔ Pages are aggressively cached.
- ➔ Data transformation code is running on getting updates from provider to produce sport feeds.
- ➔ Client apps make requests through CDN but occasionally pass to AEM.
- ➔ Easy integration.
- ➔ Relatively small infrastructure.
- ➔ Easy deployment.
- ➔ Java implementation.

Sir, we are under attack!





```
<?xml version="1.0">
<SportFeed type="Leaderboard">
  <Holes>
    <Hole id="1" toPar="5" />
    <Hole id="2" toPar="4" />
    <Hole id="3" toPar="3" />
    ...
  </Holes>
  <Leaderboard>
    <Player id="30925" name="Dustin" ...
```

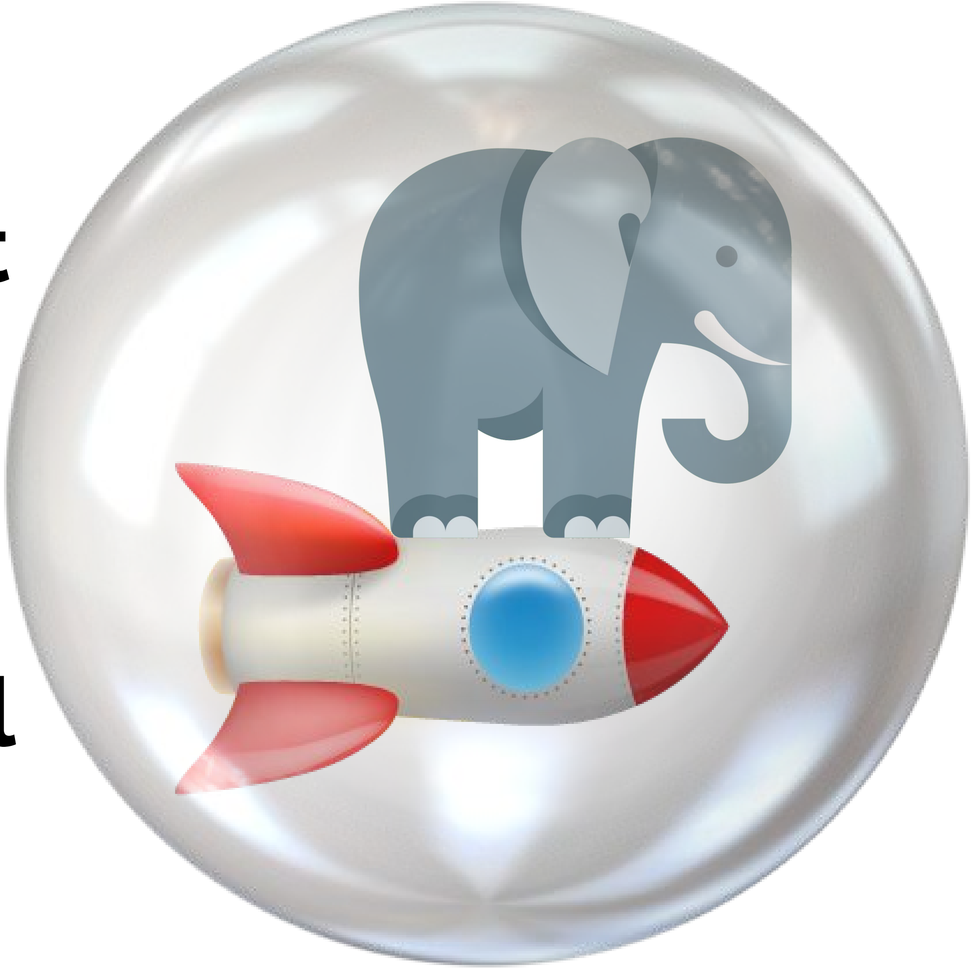
```
<?xml version="1.0">
<SportFeed type="Strokes">
  <Holes>
    <Hole id="1">
      <Score average="4" />
    </Hole>
    ...
  </Holes>
  <Strokes>
    <Stroke player="30925" hole="3" ...
```

```
<?xml version="1.0">
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soa
p/envelope/">
  <soap:Body>
    <getQualifyingStatsResponse
xmlns="http://qualifying.provider.com/ws">
      <item type="holeStats">
        <item type="holeStat" stat="putts"
holeId="1">35</item>
```

Results

- ➔ Cache invalidation became too complicated.
- ➔ Client apps work slow. UX is bad.
- ➔ Maintaining of wrong sport data takes much effort and it is not possible for not AEM developers.
- ➔ To save consistency across different client platforms additional business logic and serialization is needed.
- ➔ As soon as the number of third-party providers grows AEM performs redundant work which leads to high consumption of RAM. Client apps request for updates every 20 seconds.

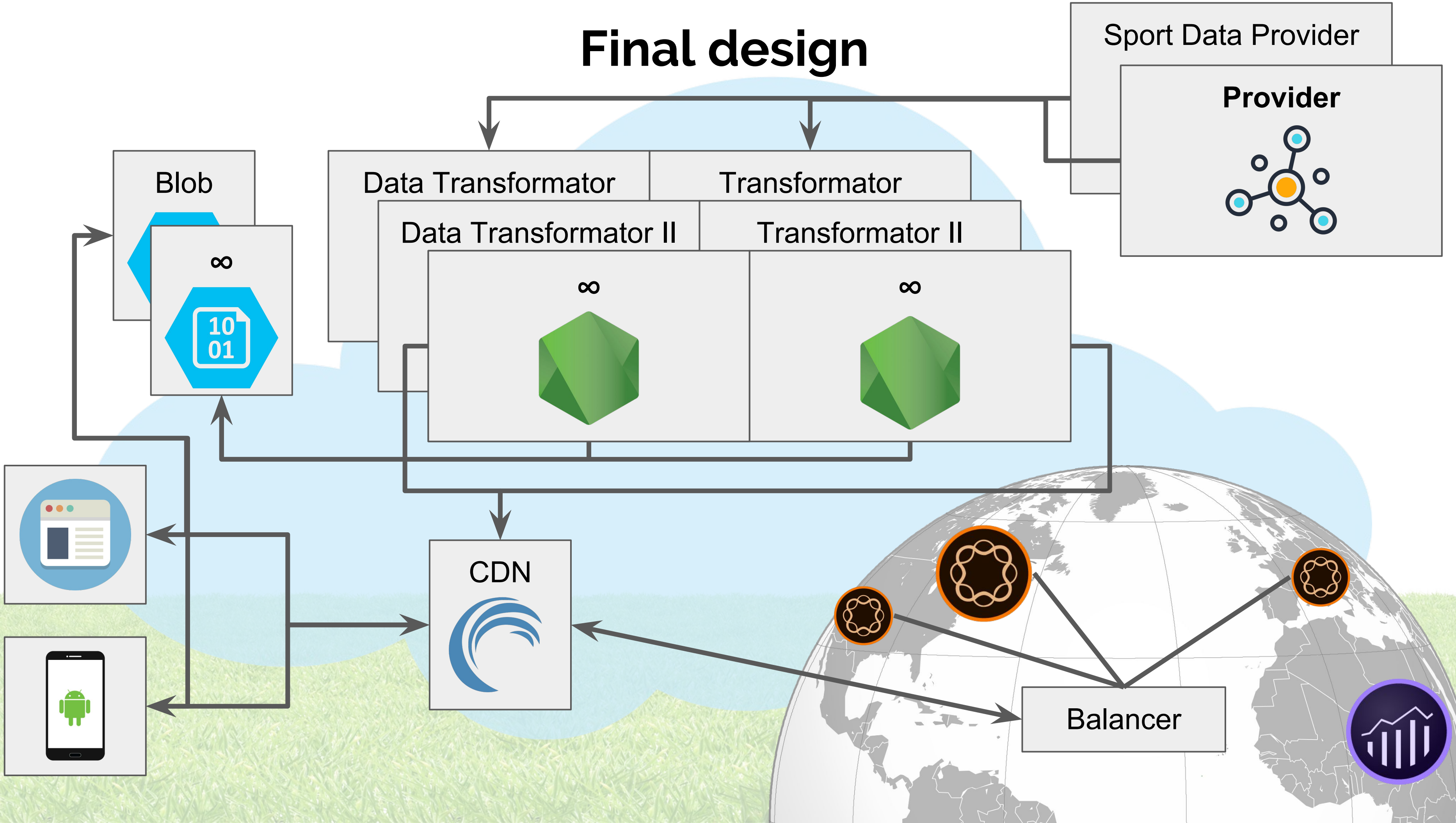
Even elephant on a rocket within spherical vacuum



can not help!

«Zed is not dead, baby»

Final design



Microsoft Azure is cloud platform

- Hybrid cloud solution which incorporates virtual machines, databases, isolated computing units (app services/functions) etc.
- Has DevOps friendly interface.
- All services use internal network.

App functions are computing units which can be written in major languages (e.g. Java, .NET, JS) and can be configured to run triggered by request, timer, socket connection etc. Those functions are highly scalable and configurable. Deployment is easy and convenient.

App function example

```
const DataTransformer = require('./transformers');

module.exports = async (context, inputDocument) => {
  try {
    const xmlTransformer = DataTransformer.createXmlTransformer();
    const metaData = xmlTransformer.getMetaData(inputDocument);
    let outputDocument = '';

    context.log.info('Transforming xml document', metaData);

    outputDocument = await xmlTransformer.process(inputDocument);
    context.outputDocument = outputDocument;

    context.done()
  } catch (exception) {
    context.done(exception)
  }
}
```

Features

- ➔ All sport related business logic and transformation are isolated in well-scalable app functions.
- ➔ All computed results are stored in a fastest blob storage which uses wide distributed CDN.
- ➔ All requests go through CDN. If updated content is needed balancer will catch request first.
- ➔ Necessary AEM data for computations will also be fetched using CDN.
- ➔ AEM instances of different size are distributed across various geo regions.

Results

- ➔ Using CDNs for every part of the project all **1’000k** requests didn’t hit servers.
- ➔ Using scalable farm of computable units in cloud allows to perform hard work out of AEM.
- ➔ All files in blob can be edited even in notepad.
- ➔ Scalability works in automatic or manual mode.
- ➔ ASAP-fixing is available for DMs/PMs.
- ➔ With help of Adobe Analytics it’s possible to configure proper AEM instances per geo region.

Problems and solutions

➔ Problem

Every client looks for a feed in specific format/structure.

Differences on the latest stages of the project can lead to missing deadline. In sport it’s an uncoverable severe fault.

➔ Solution = SSOT*

Use well-defined scheme which describes every single field and presentation form.

For instance Swagger provides convenient APIs and generators.

Making the schema more universal is a good point. Let client app render fields using predefined formats which it understands.

➔ Problem

Slow page load.

Too many resources are loaded not regarding real usage of them.

Having many libraries as a separate file wastes time for parsing and loading.

Initial rendered JSON data for JS framework can hang up AEM and browser in case of huge size.

➔ Solution

Split JS and CSS bundles per page. Modern and flexible bundler can help (WebPack, SystemJs, custom gulp configuration).

It’s better to convert JSON structures to a feed but render initial data relatively small.

➔ Problem

Low page performance.

Very often it's related to front end design: *memory leaks* (not unbind event handlers, zombie references), *loading everything* even if it's not visible yet, *unnecessary interaction with DOM*, *resource intensive computations on main thread.*

➔ Solution

Use modern high-performance framework which allows fine control of events/data binding and abstracts from DOM.

Try lazy loading.

Use «*Web Workers API*» or delay computations on main thread. Do not overuse observers.

➔ Problem

Sport data can have typos and malformed data.

It can occur due to many circumstances like bad weather, changed conditions (e.g. playing on other course) etc.

➔ Solution

Backup data introspectively.

PM/DM should be familiar with tools to fix data issues. So data format should be simple and self-explanatory (close to business). XML format works but object-oriented (JSON) is more convenient in most cases.

Conclusions

- ➔ Use caching for every available feed and distribute requests flow carefully.
- ➔ Data transformation should be performed in cloud using lambdas/app functions etc. Take away redundant load from AEM.
- ➔ Keep data mapping in declarative schema reflecting API (e.g. Swagger).

- Split JS and CSS per page.
- Use high-performance JS frameworks. Pay attention to incremental UI updates (Virtual DOM capable frameworks like React/Vue.js etc. can work quite effective), memory leaks, intensive computations.
- Your clients (e.g. web/mobile apps) should have only presentation logic.
- Try to simulate event data population including malformed and redundant data.

➔ Cloud infrastructure can save your life in one day.

➔ If something can go wrong it will go. Your server code should not kill server if something fails.

➔ Have convenient and handful tools which allow you to log and monitor activity in real time.

Sources

- ➔ Of course, I used wikipedia
- ➔ Moscow University Club
- ➔ TrackmanGolf.com
- ➔ GolfWeek.com
- ➔ GolfOnline.ru
- ➔ Lunka.ru

Thank you for your attention!