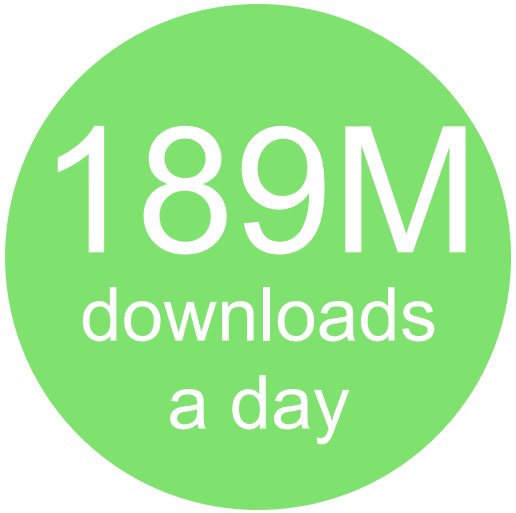
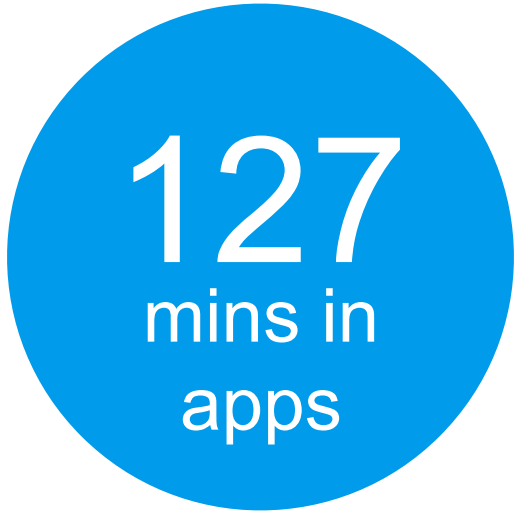
Dev Days

Connected & Disconnected Apps with Azure Mobile Apps

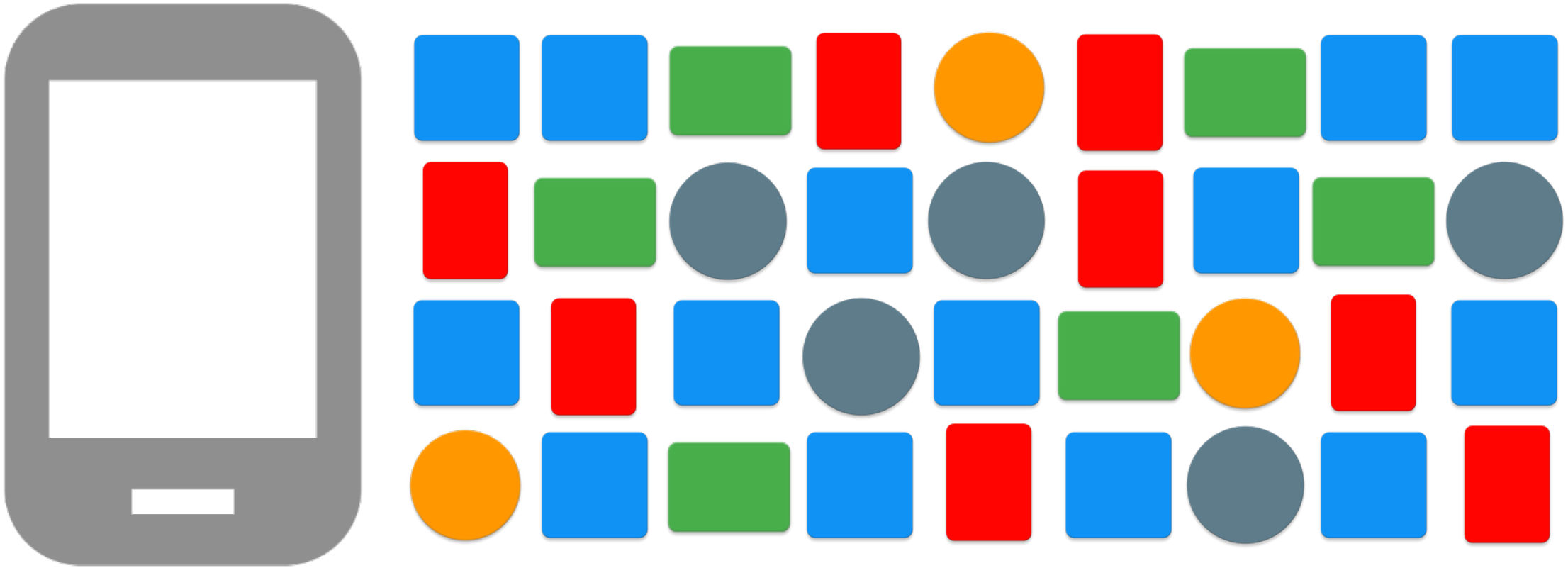
Fabio Cozzolino
@fabiocozzolino
Microsoft MVP

We ❤ Apps!

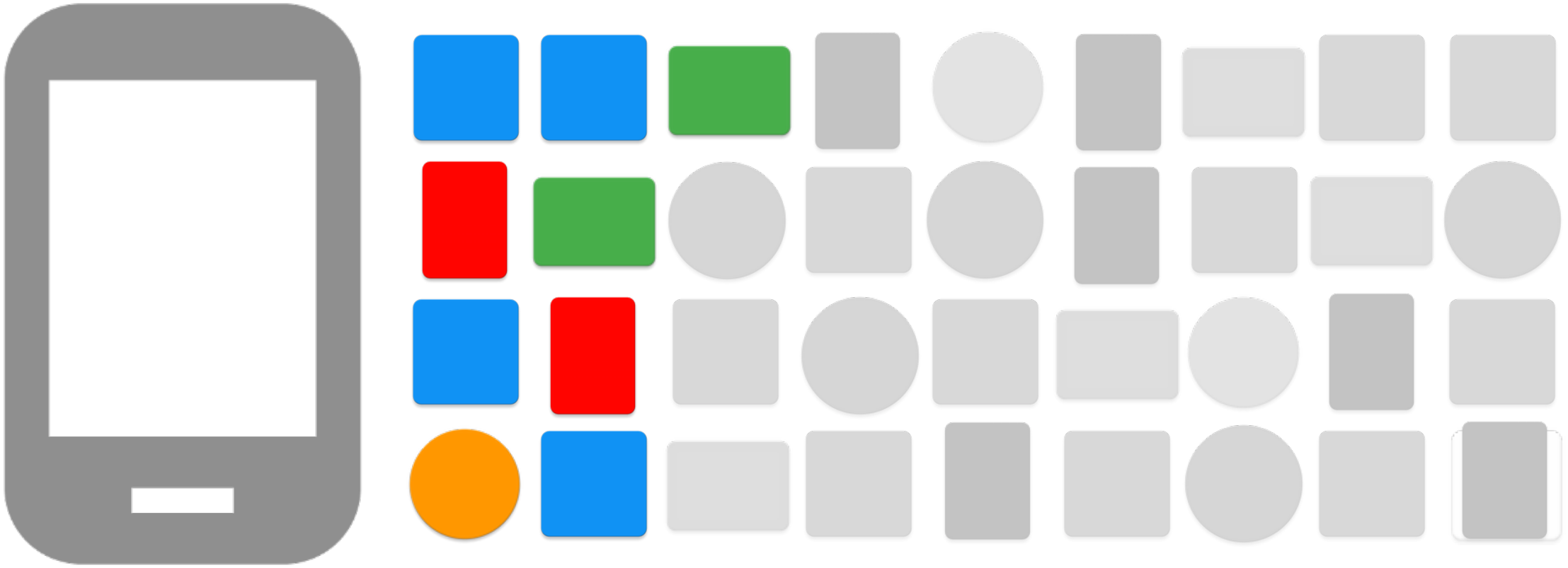




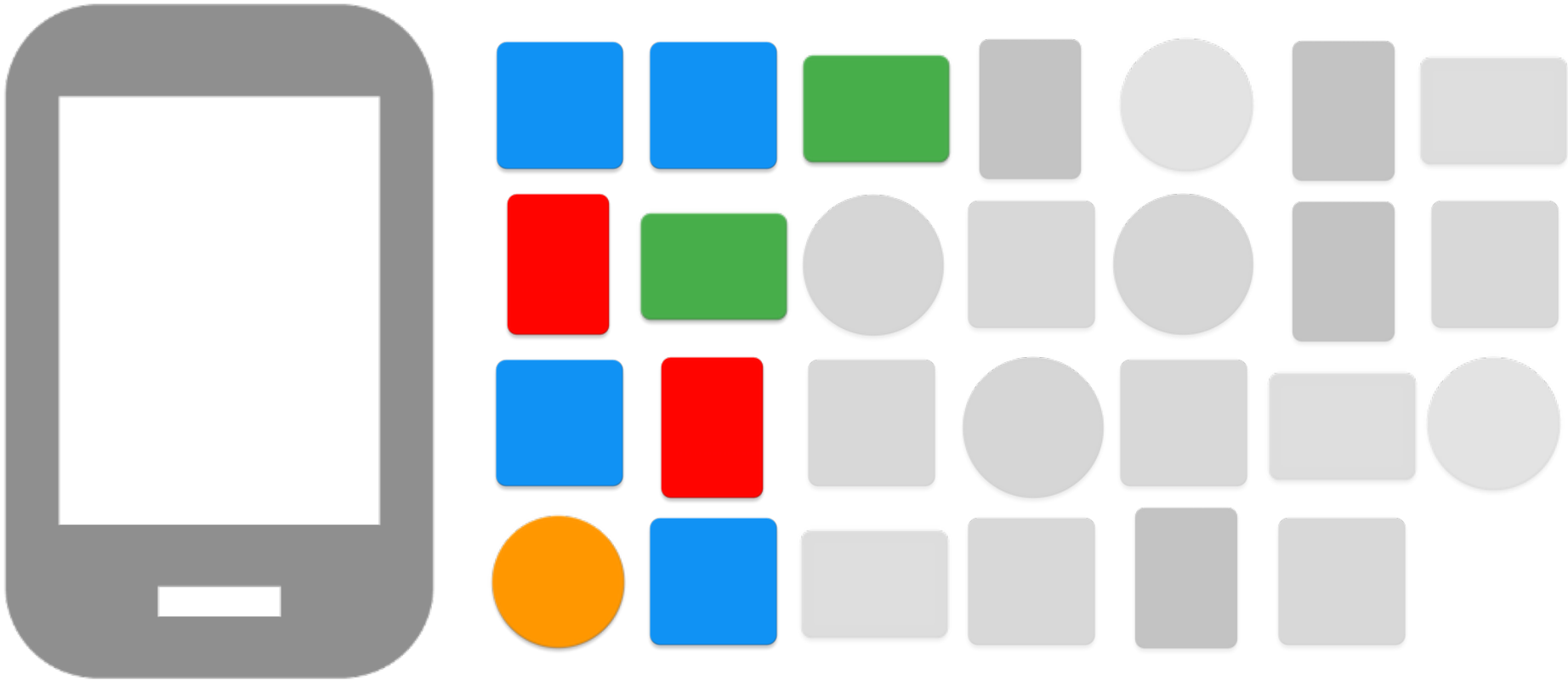
The average app user has **36** apps installed on his or her phone.



Only 1/4 are **used daily**:



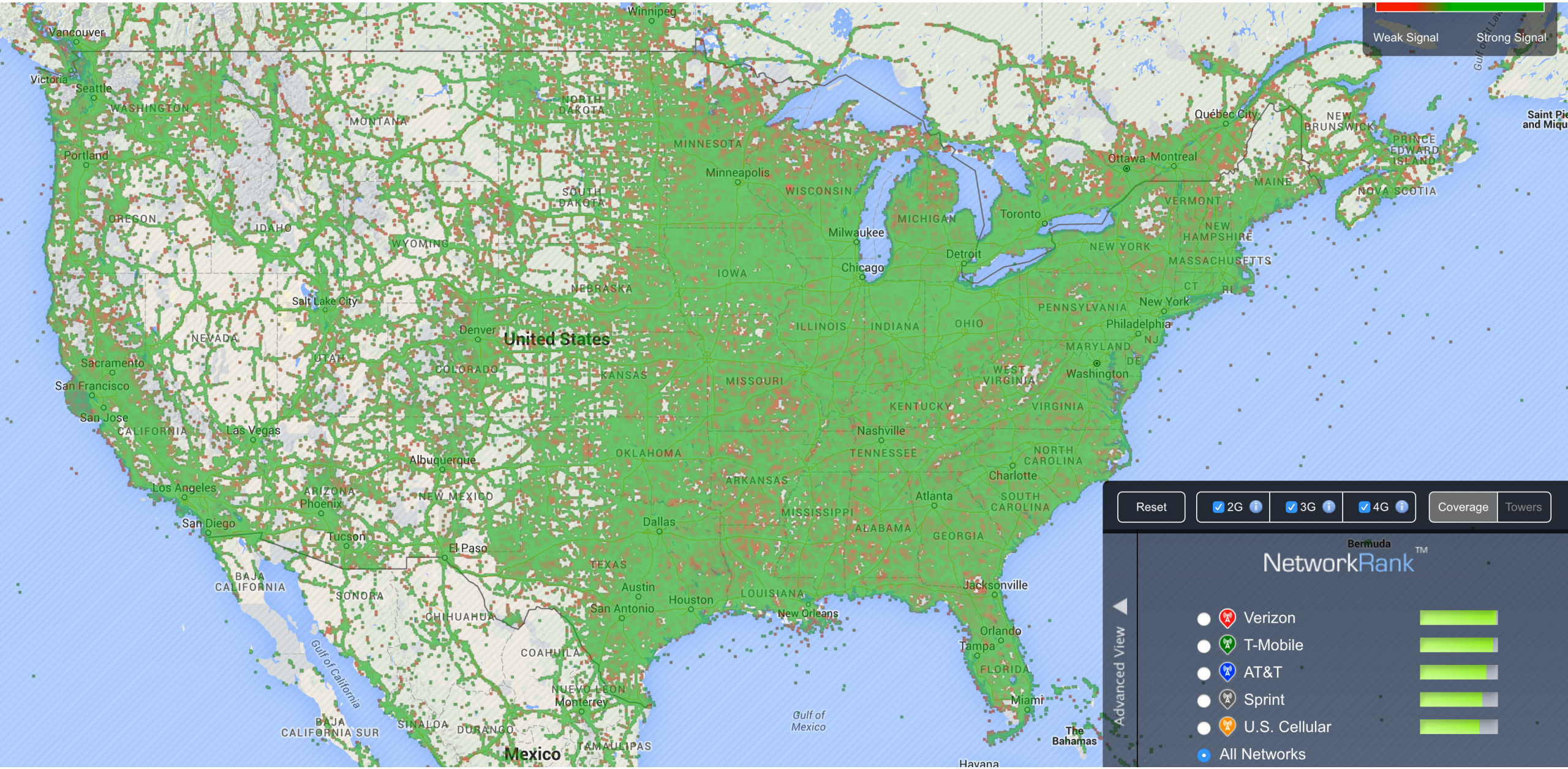
1/4 of apps are **never used**!



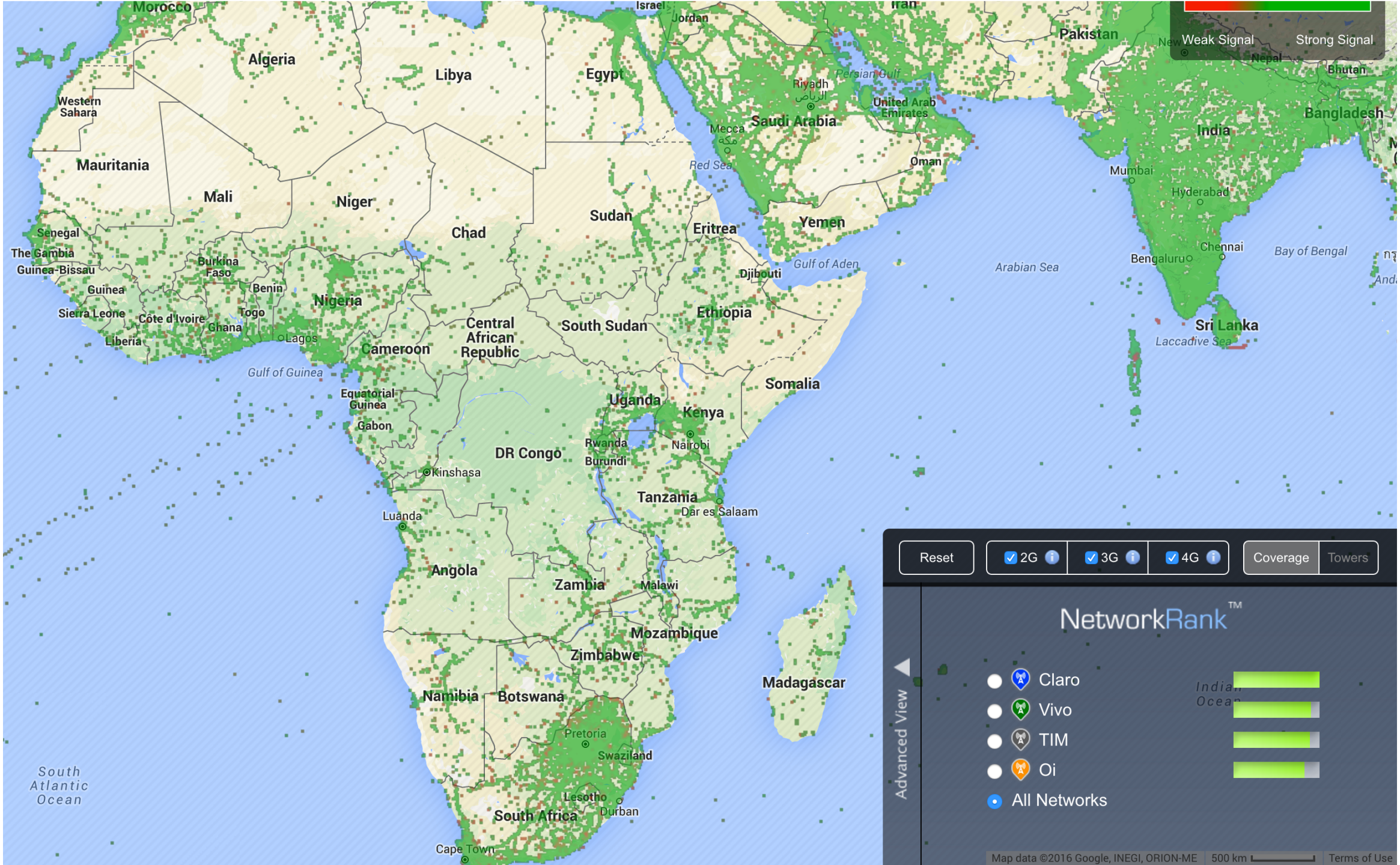
Bad App Experiences

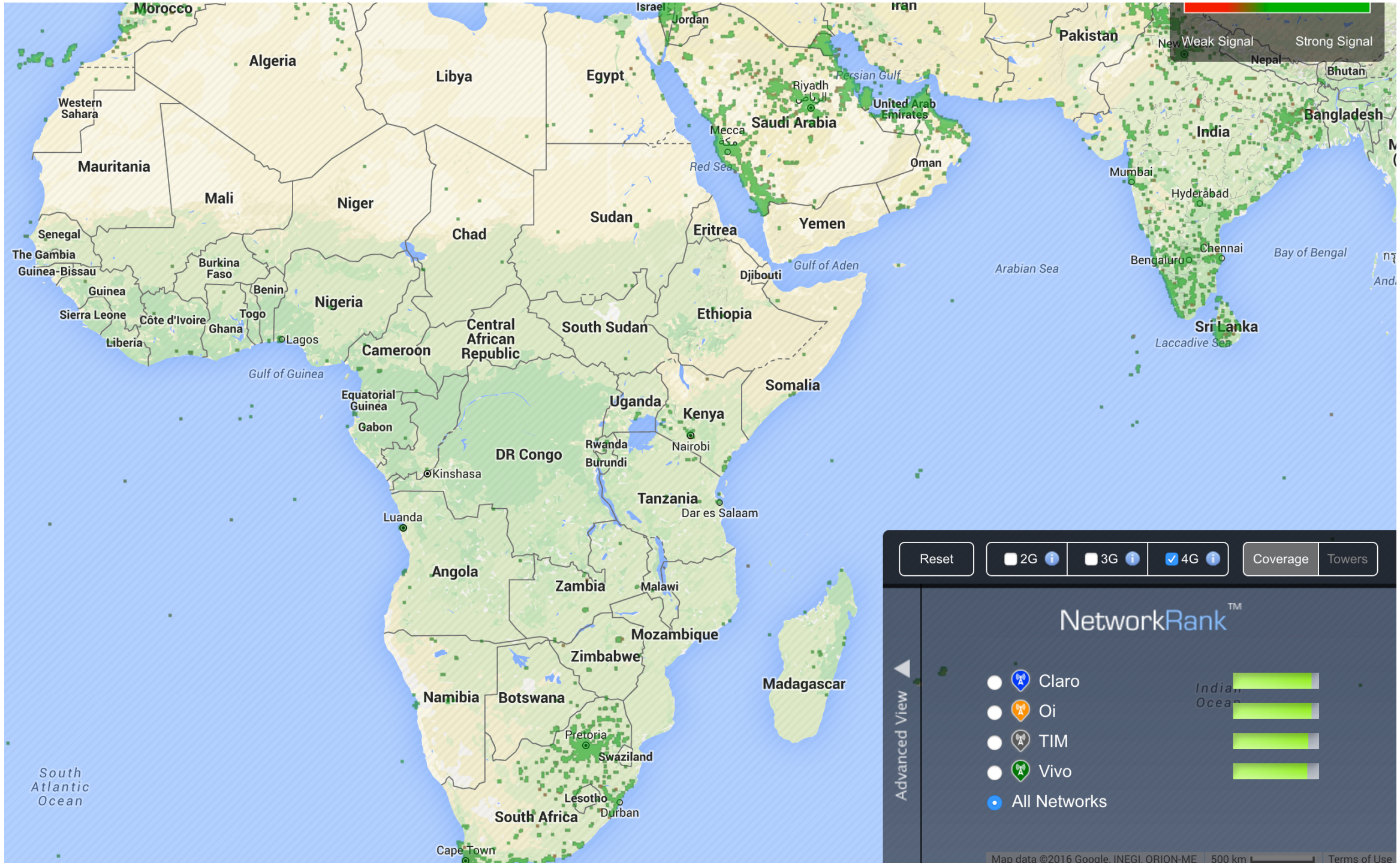
- Slow or laggy experience
- Crashes
- Not intuitive & bad user experience
- Features not as advertised
- Data not available when you need it

Always connected?



http://opensignal.com/coverage-maps


Weak Signal
Strong Signal
Reset
2G
3G
4G
Coverage
Towers
NetworkRank™
Advanced View
Claro
Vivo
TIM
Oi
All Networks
Map data ©2016 Google, INEGI, ORION-ME
500 km
Terms of Use
Morocco
Algeria
Libya
Egypt
Western Sahara
Mauritania
Mali
Niger
Chad
Sudan
Eritrea
Yemen
Saudi Arabia
Riyadh
Mecca
Red Sea
Persian Gulf
United Arab Emirates
Oman
Iran
Pakistan
India
Bangladesh
Nepal
Bhutan
Mumbai
Hyderabad
Bengaluru
Chennai
Arabian Sea
Bay of Bengal
Gulf of Aden
Djibouti
Ethiopia
Somalia
Kenya
Uganda
Rwanda
Burundi
Nairobi
South Sudan
Central African Republic
Cameroon
Nigeria
Senegal
The Gambia
Guinea-Bissau
Guinea
Sierra Leone
Liberia
Côte d'Ivoire
Ghana
Togo
Benin
Burkina Faso
Lagos
Gulf of Guinea
Equatorial Guinea
Gabon
DR Congo
Kinshasa
Tanzania
Dar es Salaam
Luanda
Angola
Zambia
Malawi
Mozambique
Zimbabwe
Namibia
Botswana
Madagascar
Pretoria
Swaziland
Lesotho
Durban
South Africa
Cape Town
Sri Lanka
Laccadive Sea
South Atlantic Ocean
Indian Ocean


Weak Signal
Strong Signal
Reset
2G
3G
4G
Coverage
Towers
Advanced View
NetworkRank™
Claro
Oi
TIM
Vivo
All Networks
Map data ©2016 Google, INEGI, ORION-ME
500 km
Terms of Use
Morocco
Algeria
Libya
Egypt
Israel
Jordan
Iran
Pakistan
Western Sahara
Mauritania
Mali
Niger
Chad
Sudan
Eritrea
Yemen
Saudi Arabia
Riyadh
Mecca
Red Sea
Persian Gulf
United Arab Emirates
Oman
India
Bangladesh
Nepal
Bhutan
Mumbai
Hyderabad
Chennai
Bengaluru
Arabian Sea
Bay of Bengal
Gulf of Aden
Djibouti
Ethiopia
Somalia
Sri Lanka
Laccadive Sea
Senegal
The Gambia
Guinea-Bissau
Guinea
Sierra Leone
Liberia
Burkina Faso
Benin
Togo
Ghana
Côte d'Ivoire
Nigeria
Lagos
Cameroon
Central African Republic
South Sudan
Gulf of Guinea
Equatorial Guinea
Gabon
Uganda
Kenya
Nairobi
Rwanda
Burundi
DR Congo
Kinshasa
Tanzania
Dar es Salaam
Luanda
Angola
Zambia
Malawi
Mozambique
Zimbabwe
Namibia
Botswana
Madagascar
Pretoria
Swaziland
Lesotho
Durban
South Africa
Cape Town
South Atlantic Ocean
Indian Ocean

Compare performance
Geneva
Limoges
Clermont-Ferrand
Lyon
Milan
Verona
Venice
Padua
Trieste
Slovenia
Zagreb
Grenoble
Turin
Croatia
Belgrade
Београд
Bosnia and
Herzegovina
Genoa
Bologna
Zadar
Craiova
Serbia
Nimes
Toulouse
Pisa
Monaco
San Marino
Sarajevo
Florence
Split
Montpellier
Marseille
Cannes
Niš
Ниш
Italy
Adriatic Sea
Montenegro
Sofia
София
Andorra
Dubrovnik
Podgorica
Kosovo
Girona
Corsica
Rome
Tirana
Macedonia
(FYROM)
Barcelona
Tarragona
Bari
Albania
Naples
Thessaloniki
Θεσσαλονίκη
Matera
Taranto
Lecce
Pompeii
Balearic Sea
Sardegna
Tyrrhenian Sea
Palma
Cagliari
Greece
Reggio
Calabria
Palermo
Patras
Πάτρα
Athens
Αθήνα
Trapani
Ionian Sea
Taormina
Marsala
Sicilia
Tunis
Algiers
NO DATA
COLLECTED
WEAK
SIGNAL
STRONG
SIGNAL
Google
Map data ©2016 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional, Mapa GISrael, ORION-ME
Terms of Use
Report a map error

What about a backend?

Plenty of Options

Azure Mobile Apps

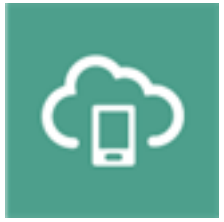
IBM MobileFirst

Amazon Web Services

SQLCipher

Couchbase

Realm



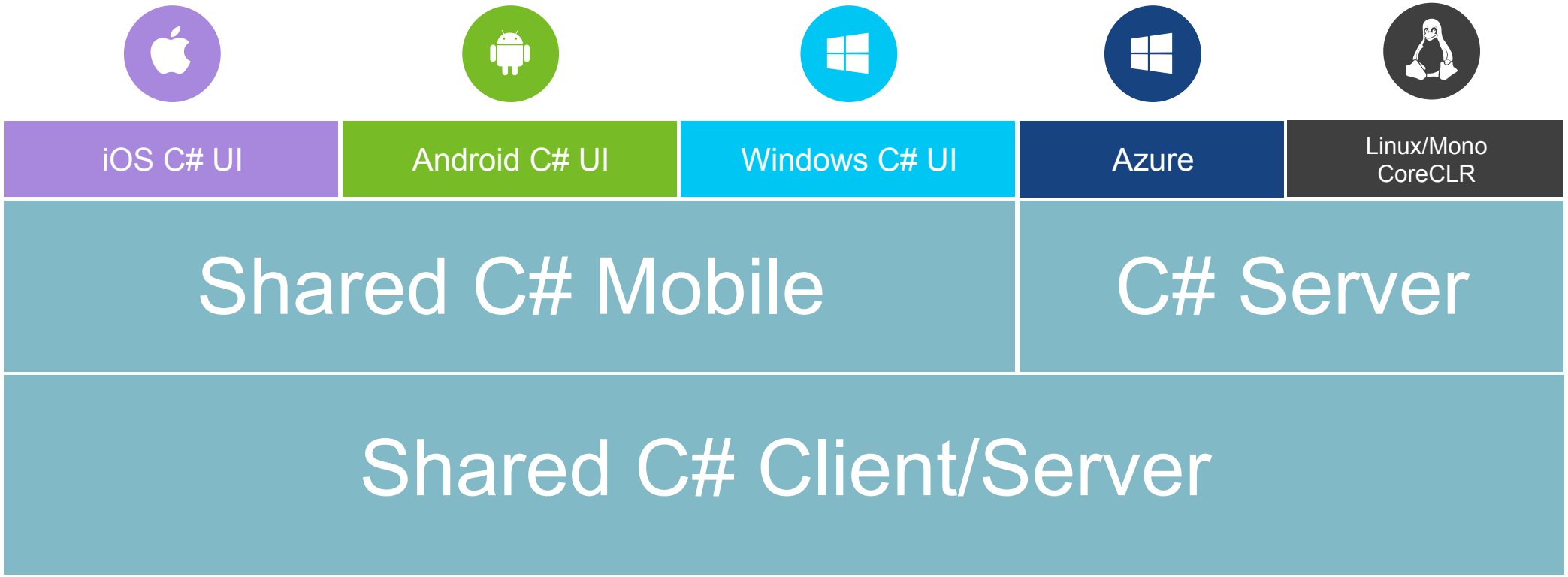
Oracle Mobile Cloud

SQLite-net

Why Azure?

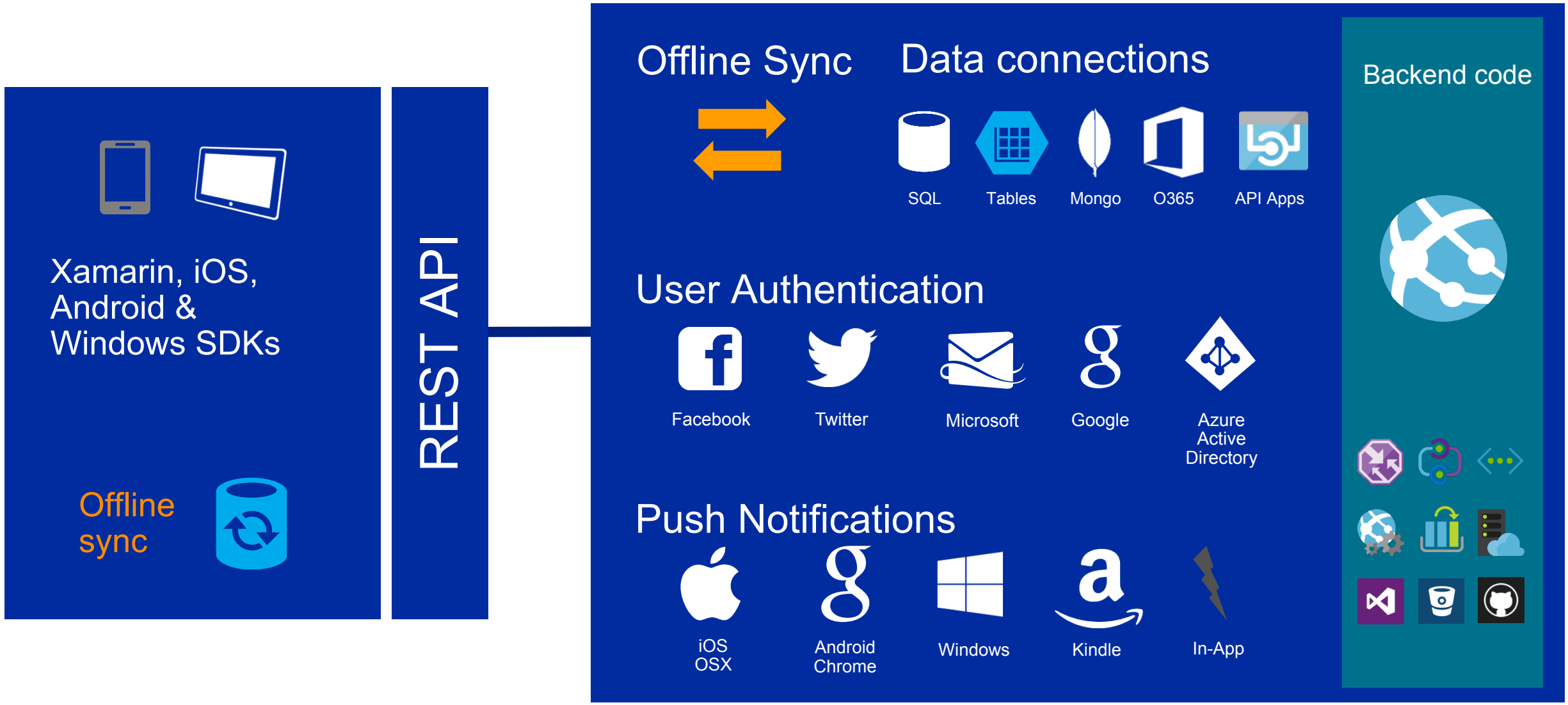
- *Extremely* powerful
- Flexible
  - Easy Tables
  - App Service
- C# SDKs available everywhere:
  - C# iOS, Android, & Windows with Xamarin
  - C# clients, written by C# developers (open source)
  - C# backend with ASP.NET

Xamarin Apps + Backend Services



Shared C# codebase • 100% native API access • High performance

Azure Mobile Apps



Create a Mobile Service

```
MobileService =  new MobileServiceClient(
    "https://myapp.azurewebsites.net");
```

Create Tables

```
IMobileServiceSyncTable<Store> table;
public async Task Init()
{
    const string path = "syncstore.db";
    var db = new MobileServiceSQLiteStore(path);
    db.DefineTable<Store>();

    var h = new MobileServiceSyncHandler();
    await MobileService.SyncContext.InitializeAsync(db, h);
    table = MobileService.GetSyncTable<Store>();

}
```

Get and Modify Data

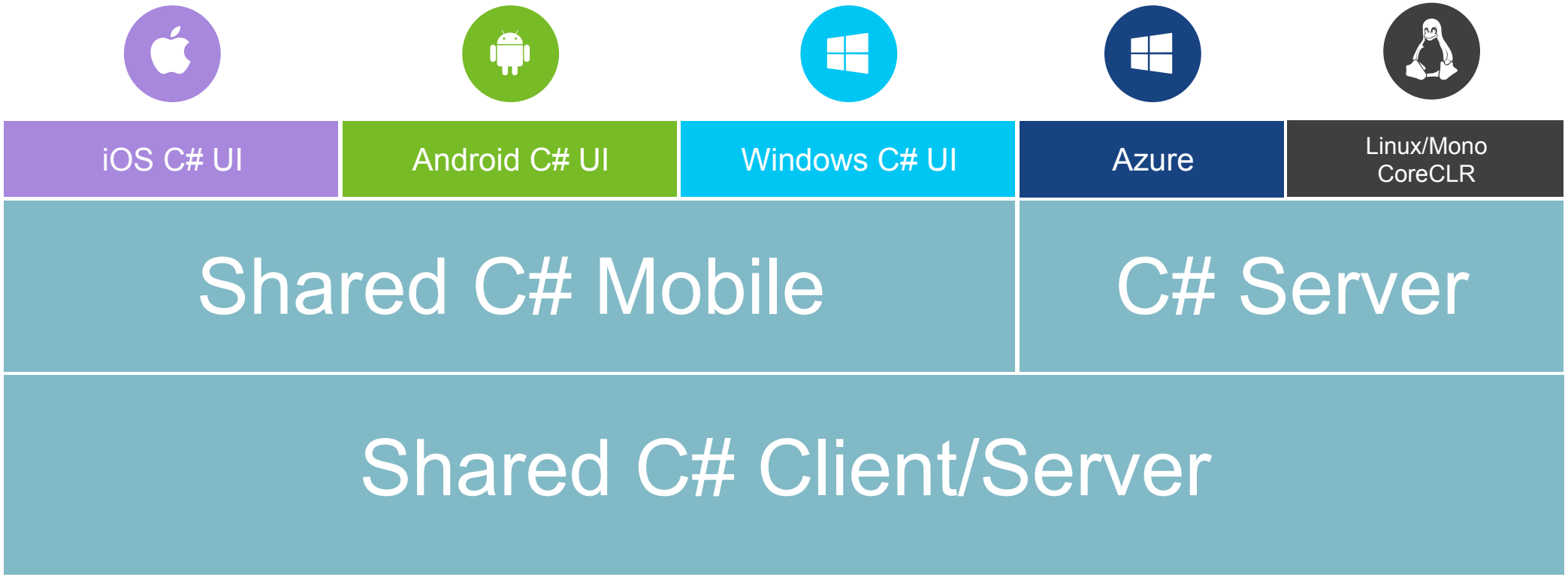
```
public async Task<IEnumerable<Store>> GetStoresAsync()
{
    await table.PullAsync("allStores", table.CreateQuery());
    return await table.ToEnumerableAsync();
}

public async Task<Store> AddStoreAsync (Store store)
{
    await table.InsertAsync (store);
    await table.PullAsync("allStores", table.CreateQuery());
    await MobileService.SyncContext.PushAsync();
    return store;
}
```

Let’s add a backend

So Much More

Mobile + Server



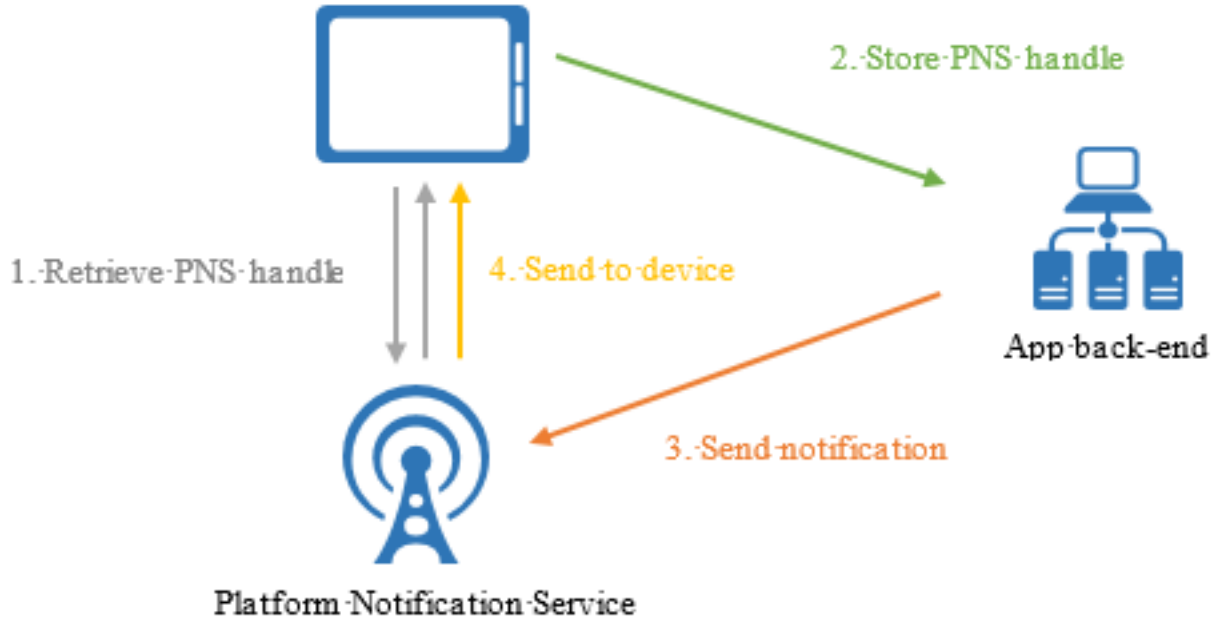
Shared C# codebase • 100% native API access • High performance

Authentication

- Rolling your own account infrastructure is difficult and time-consuming
- Secure your app with prebuilt authentication providers
  - Facebook
  - Twitter
  - Google
  - Microsoft
  - Azure AD
  - Anything OAuth 2

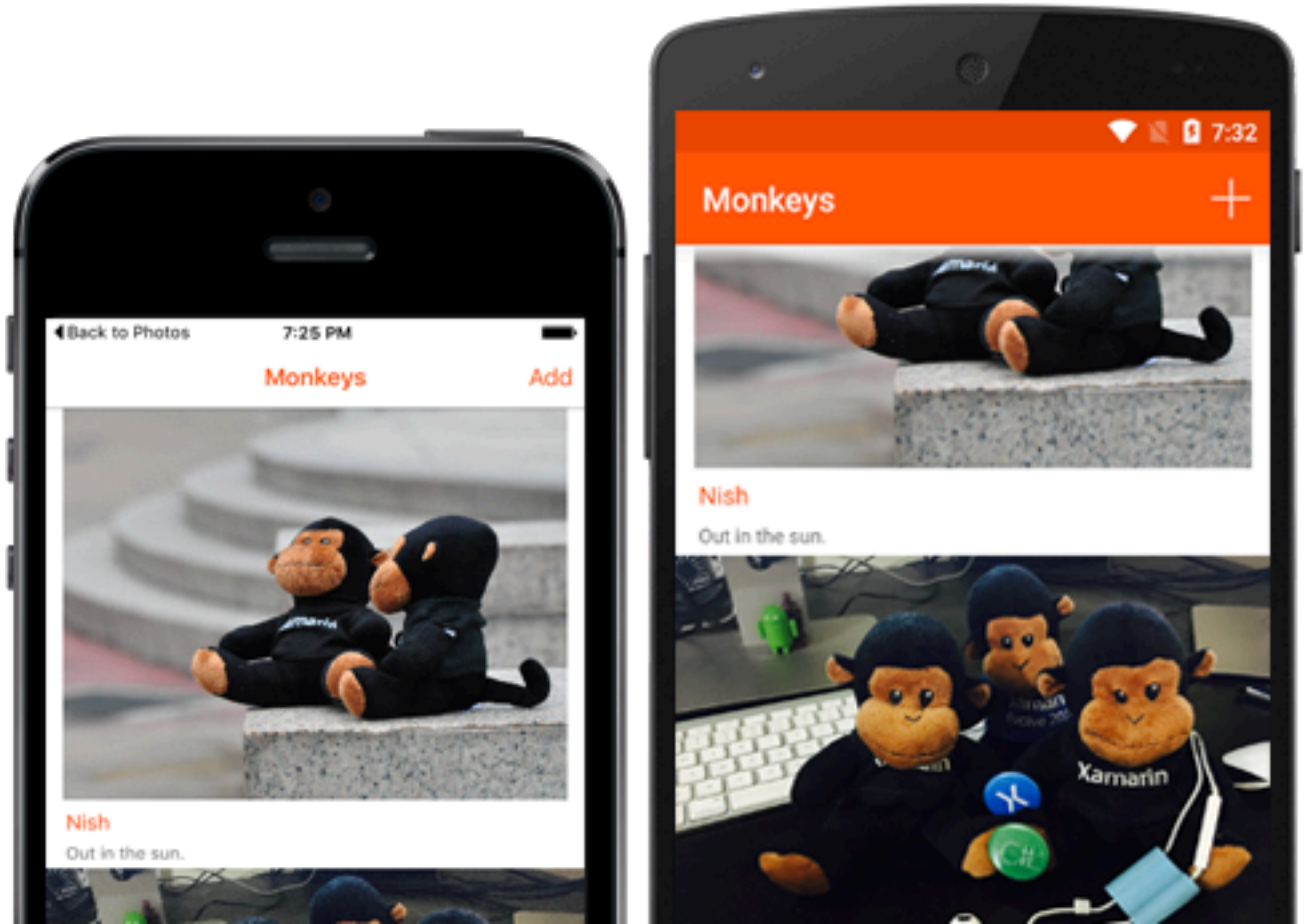
Push Notifications

- Easy-to-use, multiplatform scaled push infrastructure that allows you to send push notifications almost anywhere.



File Sync

- Sync files to Azure Storage, just like you did for structured data.



A lot of news from Connect();

Visual Studio for Mac

- Preview launched at Connect();
- Merging the Visual Studio and Xamarin Studio on Mac
  - IntelliSense and refactoring with Roslyn Compiler Platform
  - Build engine with MSBuild
  - Same debugger engines for Xamarin and .NET Core apps
  - Same designers for Xamarin.iOS and Xamarin.Android
- Based on MonoDevelop IDE and Xamarin Studio
- Still in preview… so…

Visual Studio for Mac

- A lot of new templates available
- Develop mobile applications with Xamarin.iOS, Xamarin.Android, Xamarin.Forms and Xamarin.Mac
- Develop for .NET Core apps and ASP.NET Core apps
- Templates for Connected Apps
  - Create apps with Azure Backend

Visual Studio Mobile Center

- New portal for apps
- Build, Test, Distribute and Analyze your apps
- Integrated in GitHub

Cross-Platform Development

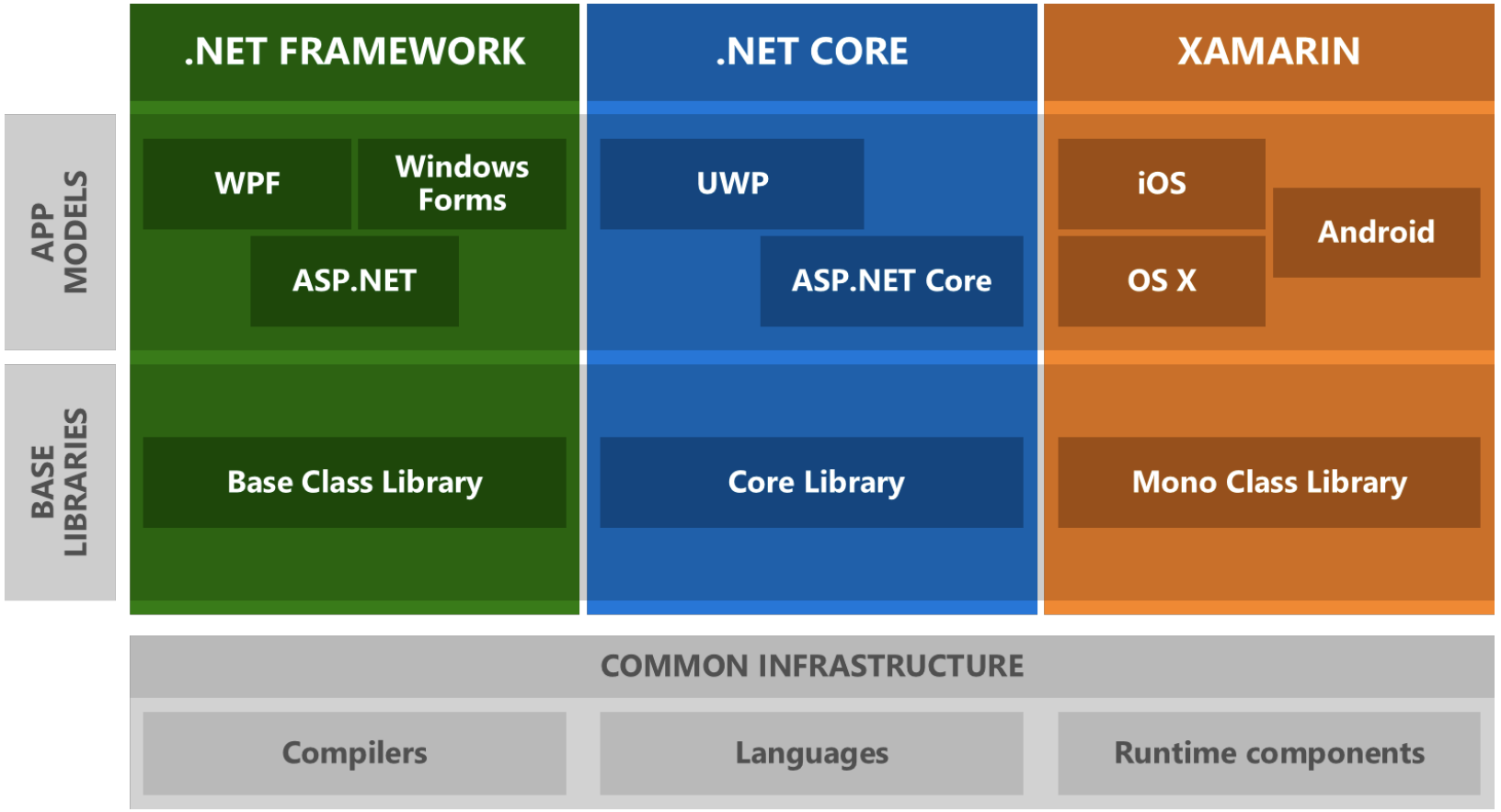
- Share code with PCL
- PCL are too much statics
  - Not ready for new platforms
- Needs something more dynamic

.NET Standard

- Defines uniform set of BCL APIs for all .NET platforms to implement, independent of workload
- Enables developers to produce portable libraries that are usable across .NET runtimes, using this same set of APIs
- Reduces and hopefully eliminates conditional compilation of shared source due to .NET APIs, only for OS APIs

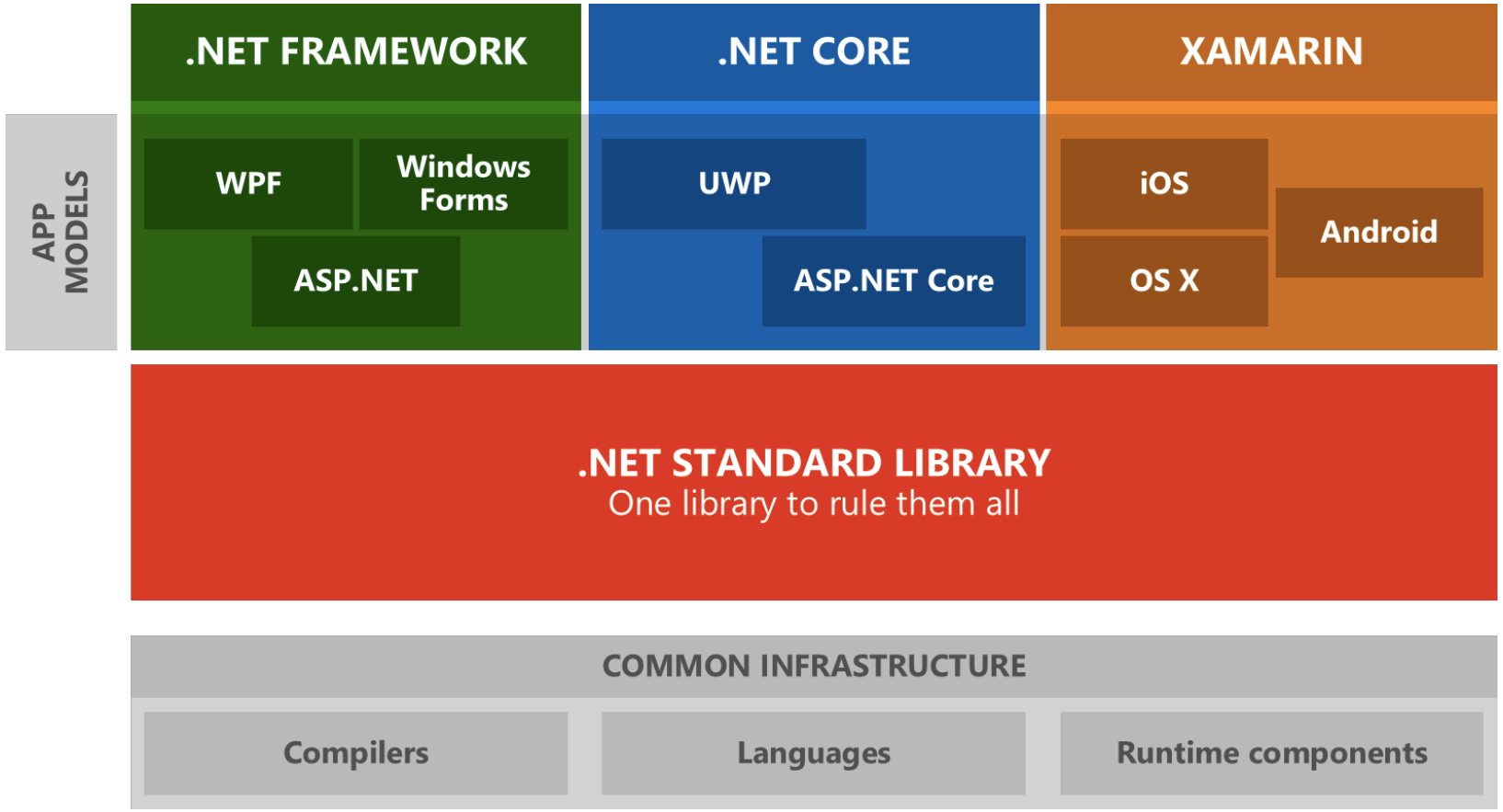
.NET Standard

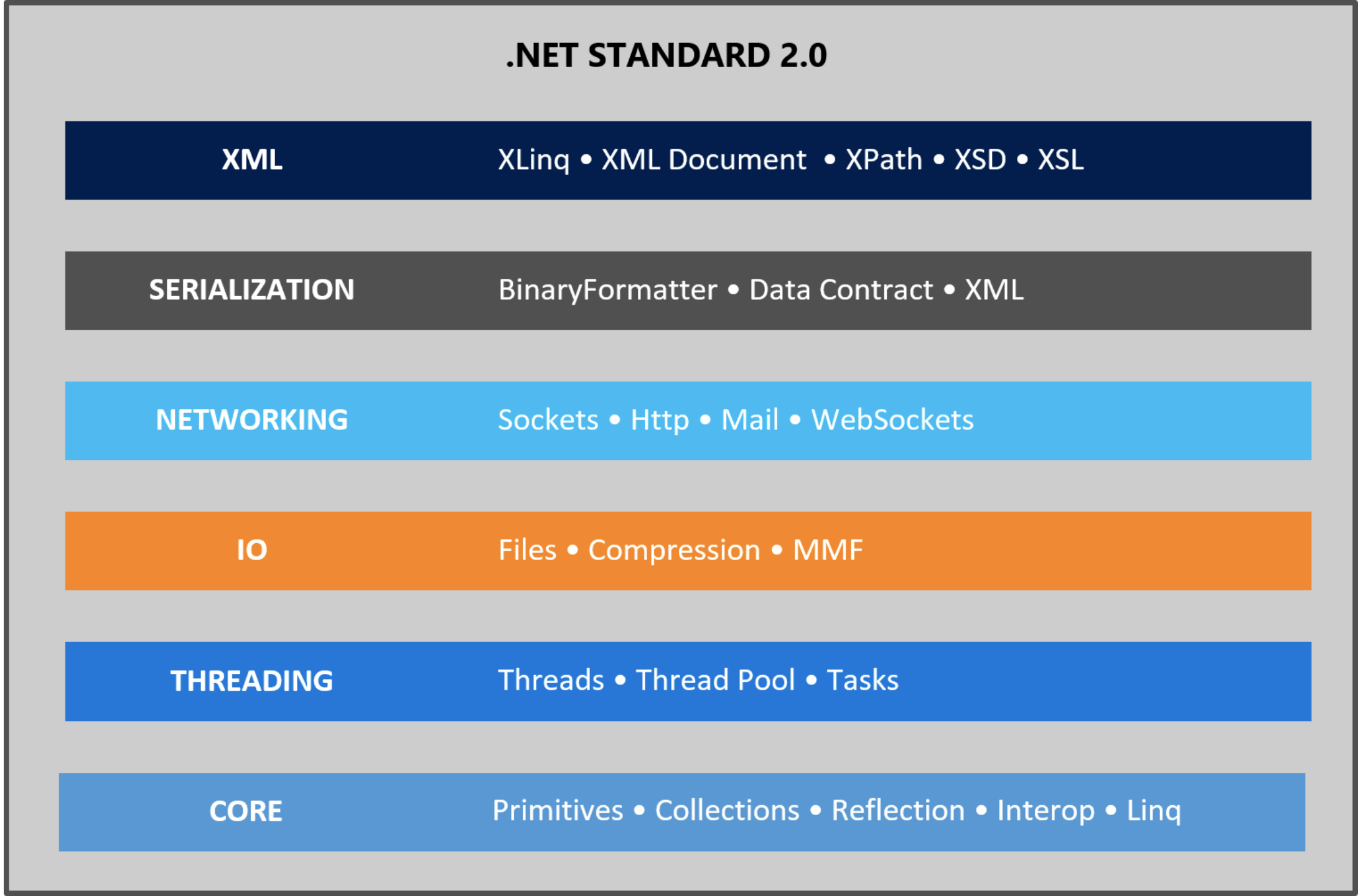
- Is a specification, not an implementation
- .NET Core is a .NET Standard implementation



.NET Standard

- Is a specification, not an implementation
- .NET Core is a .NET Standard implementation




.NET STANDARD 2.0
XML
XLinq • XML Document • XPath • XSD • XSL
SERIALIZATION
BinaryFormatter • Data Contract • XML
NETWORKING
Sockets • Http • Mail • WebSockets
IO
Files • Compression • MMF
THREADING
Threads • Thread Pool • Tasks
CORE
Primitives • Collections • Reflection • Interop • Linq

So… the PCL?

- Similarities:
  - Defines APIs that can be used for binary code sharing
- Differences:
  - The .NET Standard Library is a curated set of APIs, while PCL profiles are defined by intersections of existing platforms
  - The .NET Standard Library linearly versions, while PCL profiles do not
  - PCL profiles represents Microsoft platforms while the .NET Standard Library is agnostic to platform

Questions?

Lunch!

Fabio
Cozzolino
Microsoft MVP

fabio@dotnetside.org

fabiocozzolino.eu

@fabiocozzolino