ENT410

Data Caching and Synchronization

Download class materials from
university.xamarin.com

Microsoft

Xamarin University

Objectives

1. Determine your Connectivity Strategy
2. Cache Results from a Server
3. Synchronize to a Remote Server
4. Evaluate Data Sync Tools

Determine your Connectivity Strategy

Xamarin University

Tasks

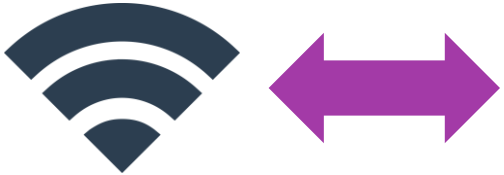
1. Make your network calls more resilient
2. Choose a suitable data strategy for your mobile application

Phones lose connectivity

- Phones will fail to connect to the network for a variety of reasons; you must program defensively around all network access code
  - Airplane mode is turned on
  - Network connectivity is lost
- Test your applications in areas where the network connectivity is poor and drops in and out such as moving such as trains, subways and tunnels

Modern app users expect offline support

- Disconnection is the rule and not the exception



Network can be affected by location, environment, however users expect to have a rich offline experience

Determining connectivity

- For network-centric apps you'll need to be able to detect network connectivity and the type of connection to perform your work

Some platforms allow you to test connectivity to a particular server

```
string HostName = ...;
bool canConnect = await CrossConnectivity.Current
                                .IsRemoteReachable (HostName);
```

APIs used for detecting network connectivity are platform-specific, but there are wrappers to provide cross-platform access

See https://github.com/jamesmontemagno/Xamarin.Plugins/tree/master/Connectivity for an example of a great plugin for Xamarin that easily adds connectivity testing

Three different types of network apps

- Most networked applications will use either a server-based approach to data, a local cache of the data, or a combination (offline editing)

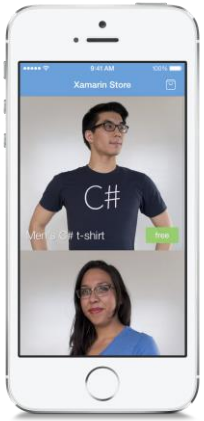
Online Only

Online with an Offline Cache

Online with Offline Editing

Online-only apps

- Online-only apps should only work when an active network connection is present such as VOIP, messaging or online banking apps



Required services can only be used when the device is online

Microsoft and Apple will test your app in airplane mode during the review process, make sure your application handles this transition gracefully

Online-only apps

- Inform the user when the application is offline and unable to retrieve data so they know the full functionality is unavailable

Summary

1. Make your network calls more resilient
2. Choose a suitable data strategy for your mobile application

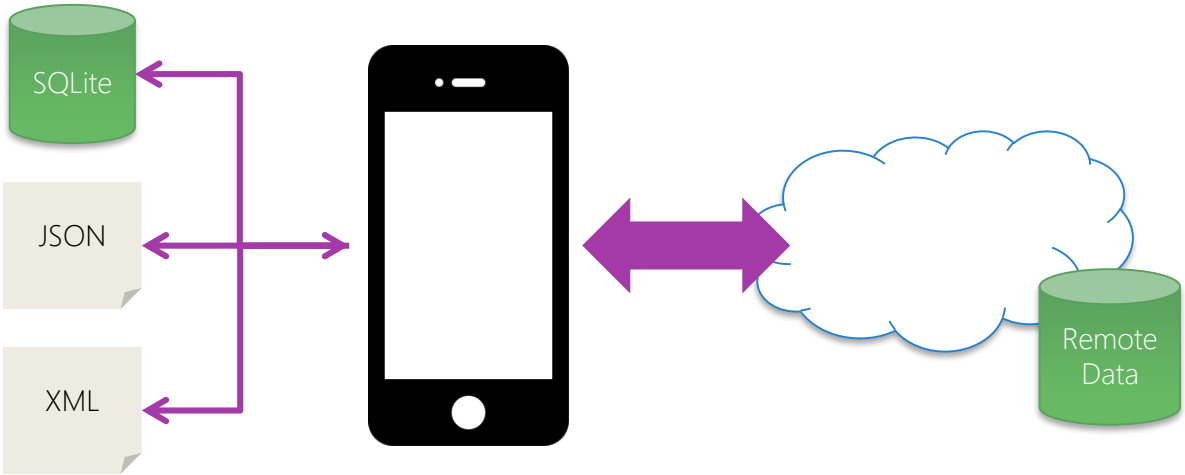
Cache Server Results

Tasks

1. Cache your data by saving network requests to the device database

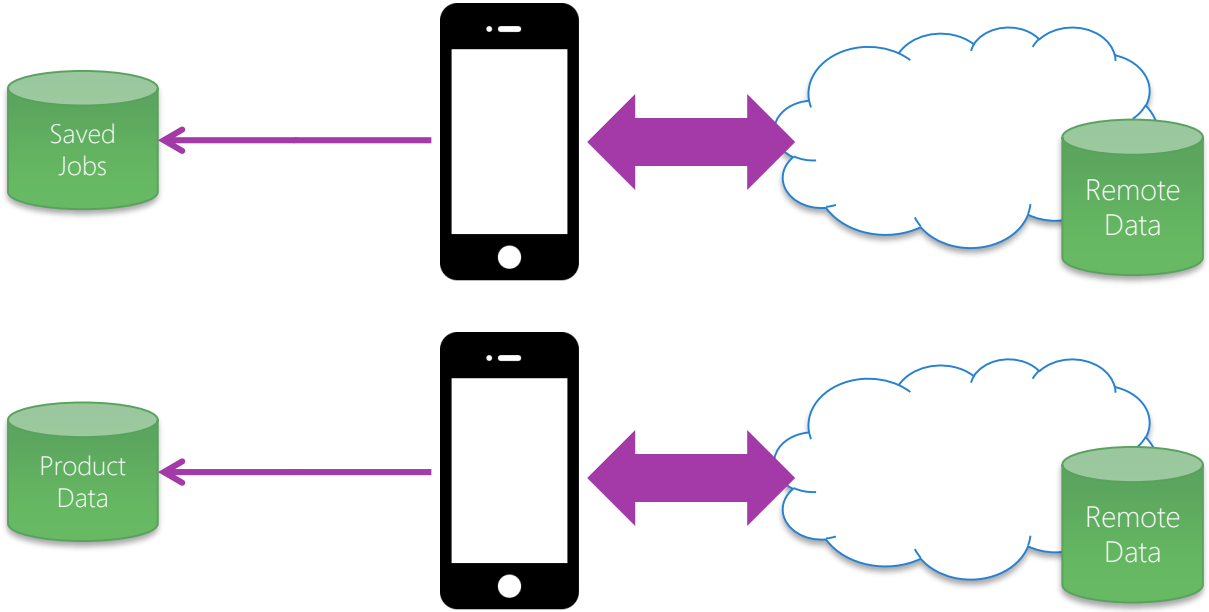
Data caching

- **Data Caching** refers to the ability to store data for later access to make subsequent access either faster or available when we cannot connect to the source of truth for the data



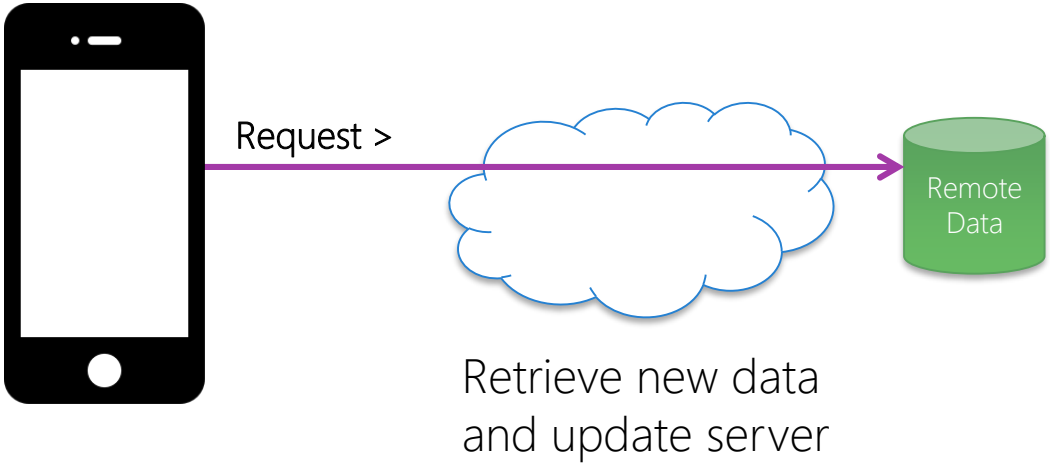
What Data to Cache

- Cache data to all your app to operate offline - depending on the needs of the application you may cache a little or a lot



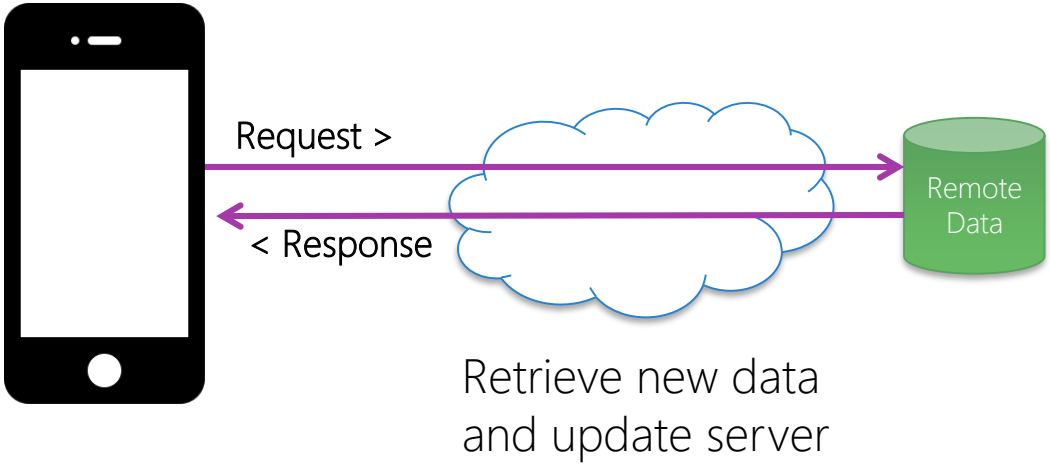
Online with local cache

- Retrieve data from online services and store it in a local cache (JSON, SQLite, etc.); use this offline data when the network is unavailable



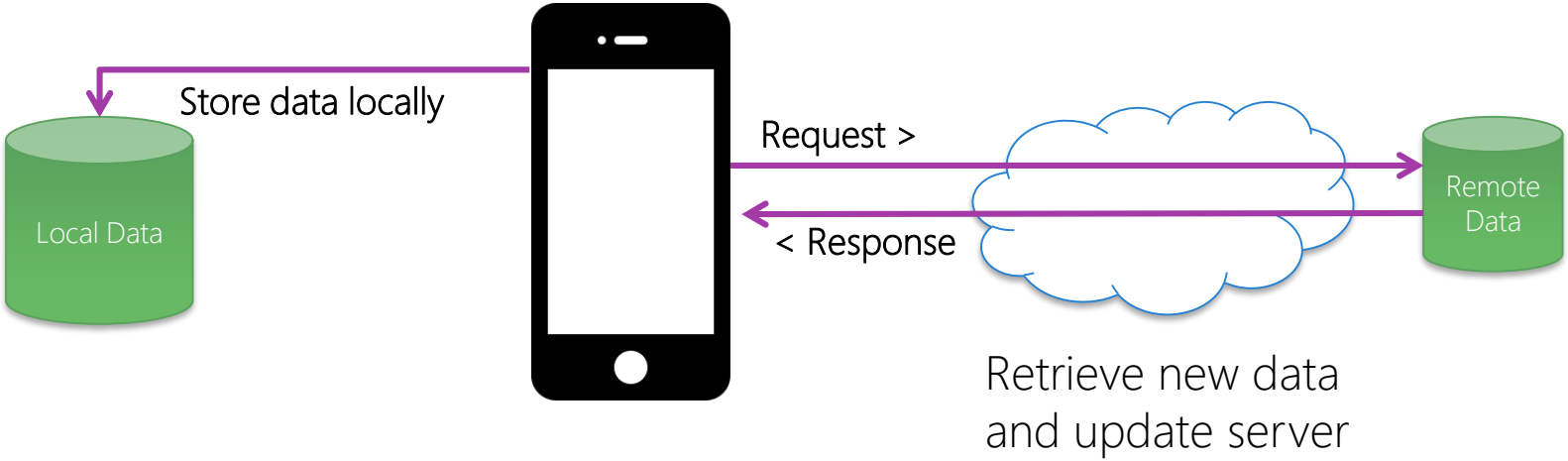
Online with local cache

- Retrieve data from online services and store it in a local cache (JSON, SQLite, etc.); use this offline data when the network is unavailable



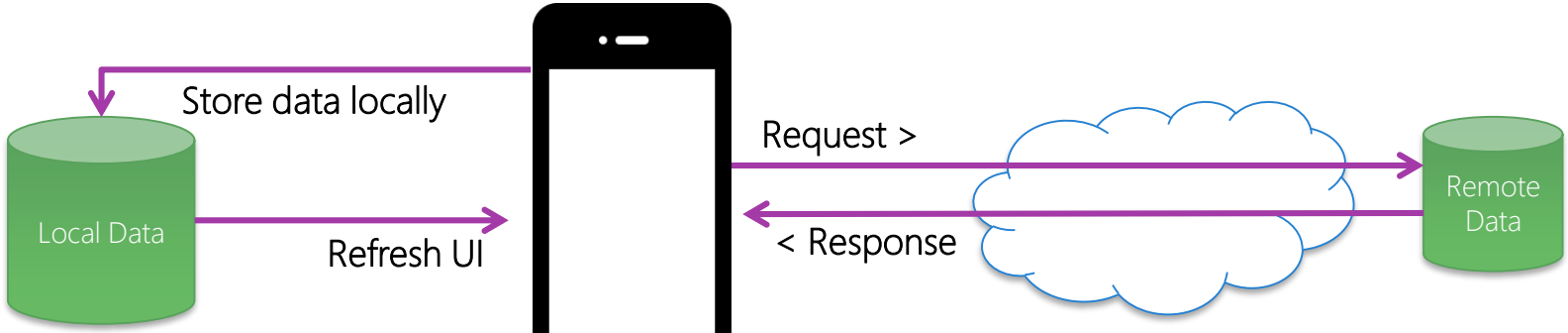
Online with local cache

- Retrieve data from online services and store it in a local cache (JSON, SQLite, etc.); use this offline data when the network is unavailable



Online with local cache

- Retrieve data from online services and store it in a local cache (JSON, SQLite, etc.); use this offline data when the network is unavailable

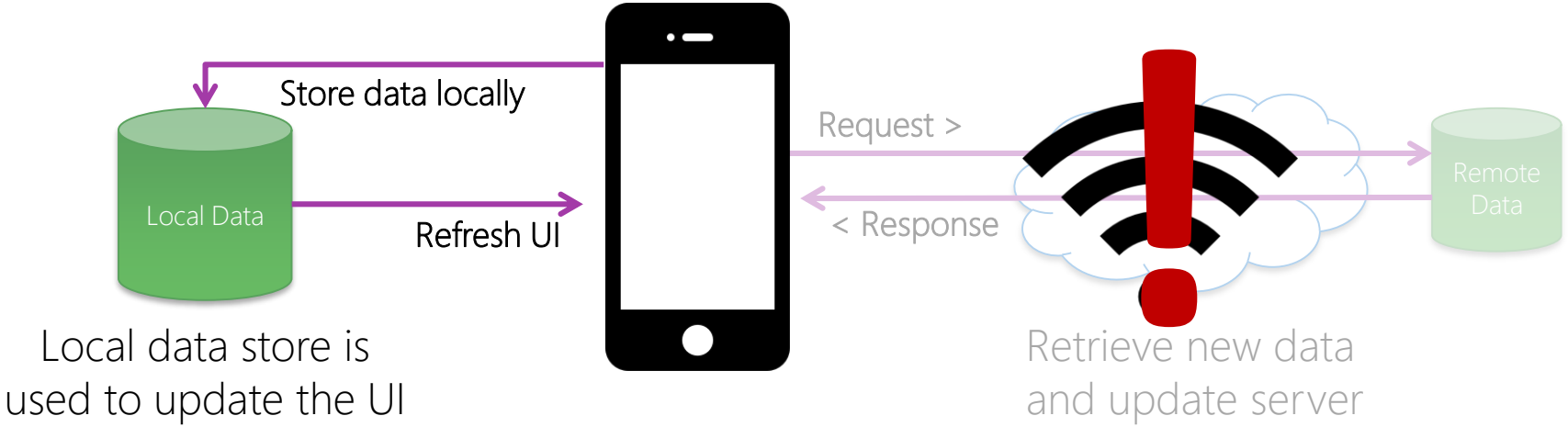


Local data store is used to update the UI

Retrieve new data and update server

Online with local cache

- Retrieve data from online services and store it in a local cache (JSON, SQLite, etc.); use this offline data when the network is unavailable



Selecting candidates for caching

- Not all service data is appropriate for caching, prefer to cache data that has a long lifespan vs. data that changes frequently

Good	Bad
▪ Static data such as locations ▪ Information that will be accessed frequently that may have a timeout assigned	▪ Frequently changed data such as weather or bank balances ▪ Time sensitive information

Coding offline support

- Refresh the data in the local cache from the remote server when it is available and have the UI view the information from the cache

```
async Task<IList<Job>> LoadPageFromNetwork()
{
   // Get the data from our service
   var service = new JobDataService();
   var data = await service.GetJobsForSearch(LastSearch, CurrentPage);
   ...
   // Store the jobs in the local database for online/offline use
   await App.DataManager.CoreDatabase.StoreJobs(data);
   return data;
}
```

Caching Assets

- Assets should be stored with cached data; **`UriImageCache`** default cache time can be extended image assets in Xamarin.Forms apps

```
public class ImageCacheConverter : IValueConverter
{
    public int DaysToCache { get; set; } = 30;
    public object Convert (object value, Type targetType, object parameter,
          CultureInfo culture)
    {
        return new UriImageSource {
            Uri = new Uri(value.ToString ()),
            CachingEnabled = true,
            CacheValidity = new TimeSpan(DaysToCache, 0, 0, 0, 0)
        };
    }
    ...
}
```

Flash Quiz

Xamarin University

Flash Quiz

① What can cause a phone to lose network connectivity (Choose all that apply)

a) Poor reception

b) Wi-Fi network change

c) Airplane mode

d) All of the above

Flash Quiz

① What can cause a phone to lose network connectivity (Choose all that apply)

a) Poor reception

b) Wi-Fi network change

c) Airplane mode

d) All of the above

Flash Quiz

② When you find you can establish a connection to a server, you don't need to worry about network failure

a) True

b) False

Flash Quiz

② When you find you can establish a connection to a server, you don't need to worry about network failure

a) True

b) False

Individual Exercise

Caching Downloaded Data

Xamarin University

Summary

1. Cache your data by saving network requests to the device database

Synchronize to a Remote Server

Tasks

1. Discuss and establish possible patterns for offline editing synchronization
2. Examine challenges with Synchronization

Best Practices

Claire Moss
@aclairefication

Follow

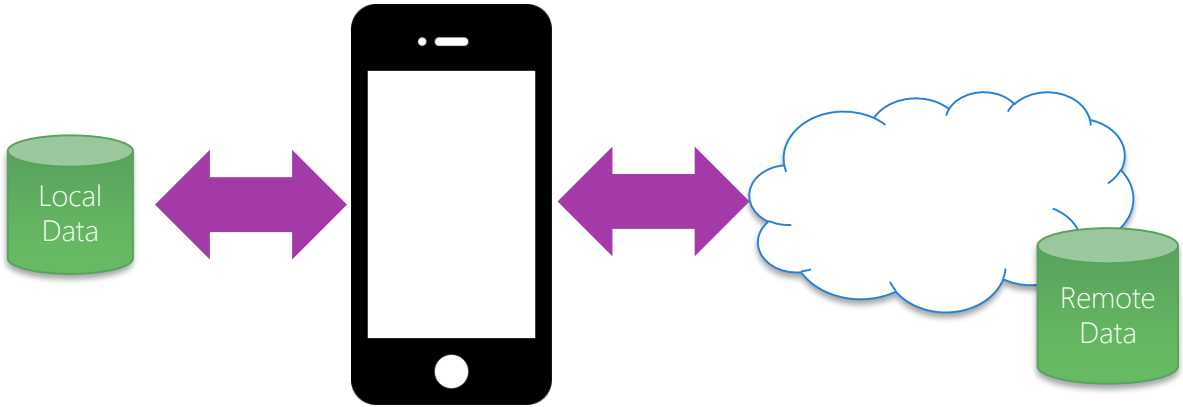
There are no best practices. Only good practices in context. @LMaccherone @t_magennis #AgileMetrics #DataScience

11:57 PM - 24 Feb 2015

8 RETWEETS **3** FAVORITES

Data Synchronization

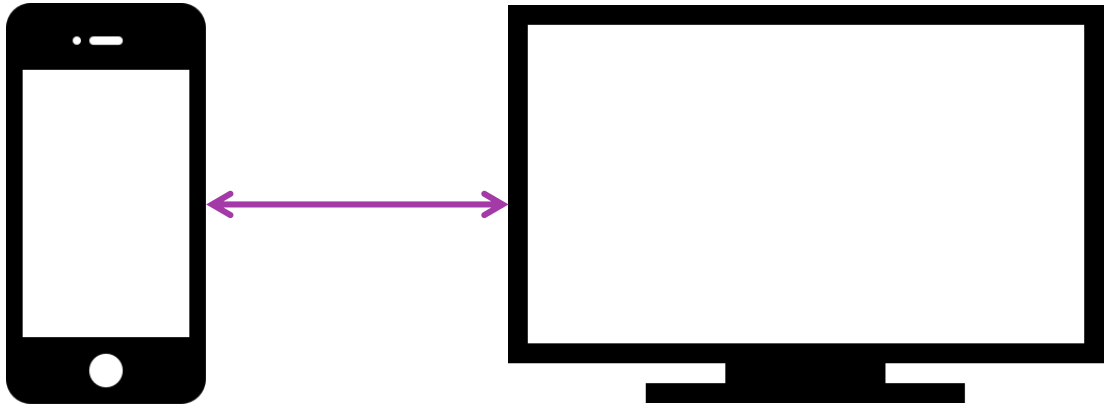
- Data Synchronization refers to the ability to make changes on a local device and then have the changes merged to a remote 'source of truth' database so the two systems appear to have identical data



The Client and Server need each other

- In order to co-ordinate the syncing activities between the server and client, ideally you'll be coding both the server and the client

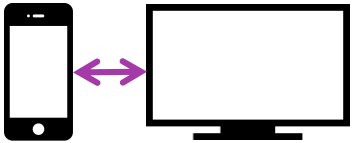
Xamarin.iOS
Xamarin.Android
Windows



Web Api
ASMX
WCF
Service Stack

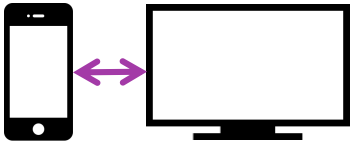
Having control of the server is required to do sync *well*. If you don't have control of the server you may have to do additional work.

Reasons for offline editing

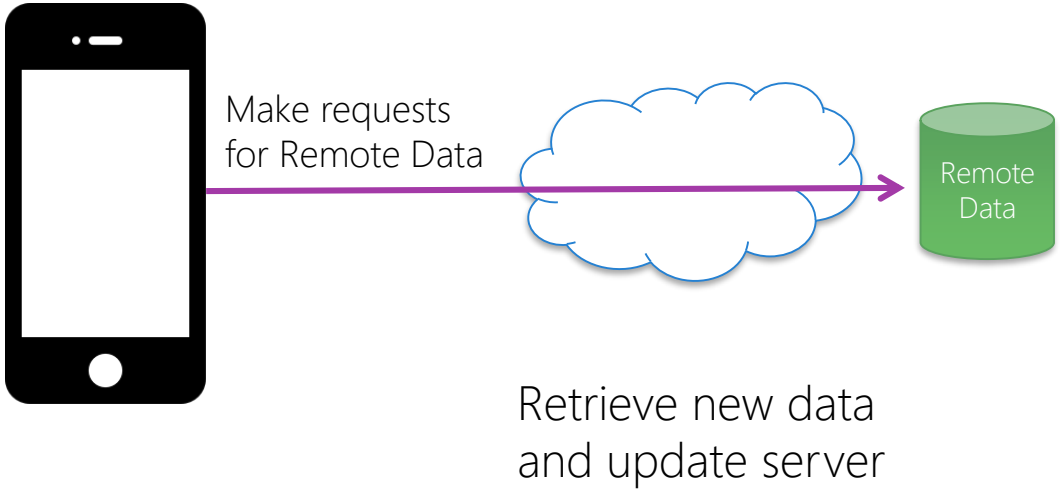


- Many applications require the use of editing data for sending to a server for later processing, especially for data collection in areas of poor connectivity

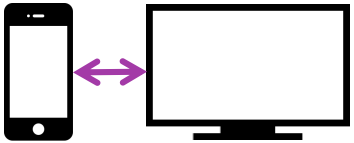
Offline editing and synchronization



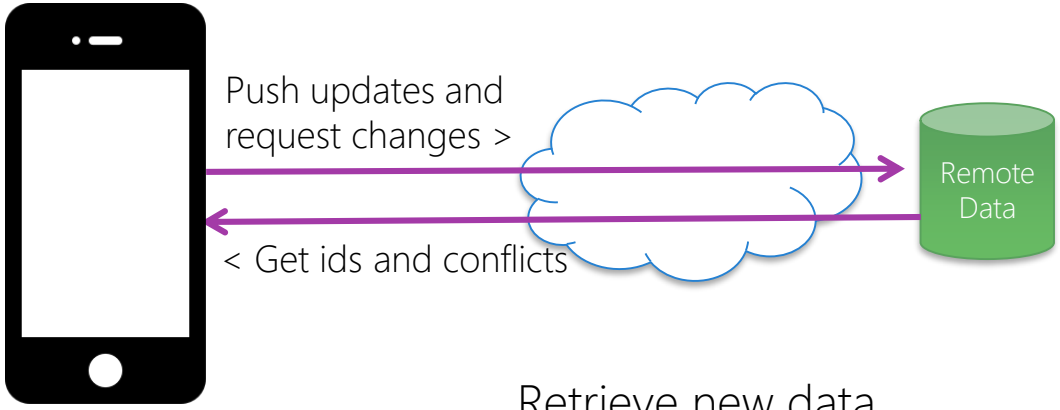
- Most networked application will use either a server based approach to data, a local cache of the data or the ability to work offline with both



Offline editing and synchronization

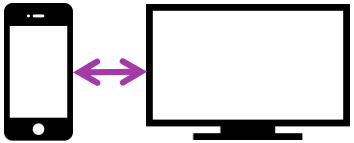


- Most networked application will use either a server based approach to data, a local cache of the data or the ability to work offline with both

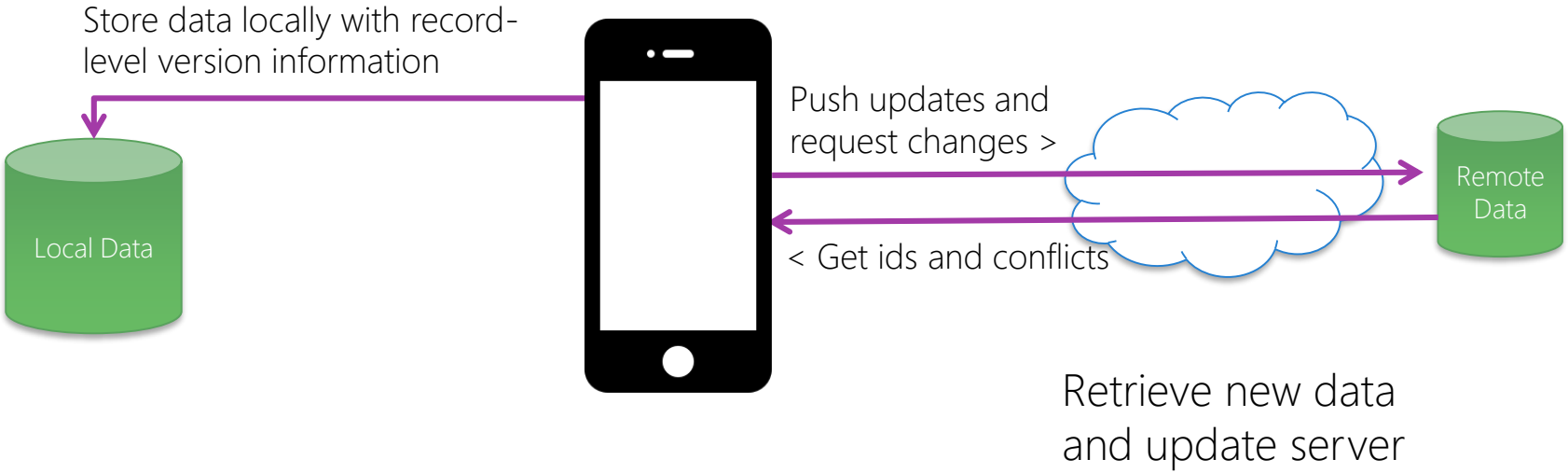


Retrieve new data
and update server

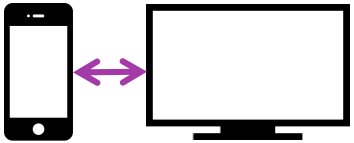
Offline editing and synchronization



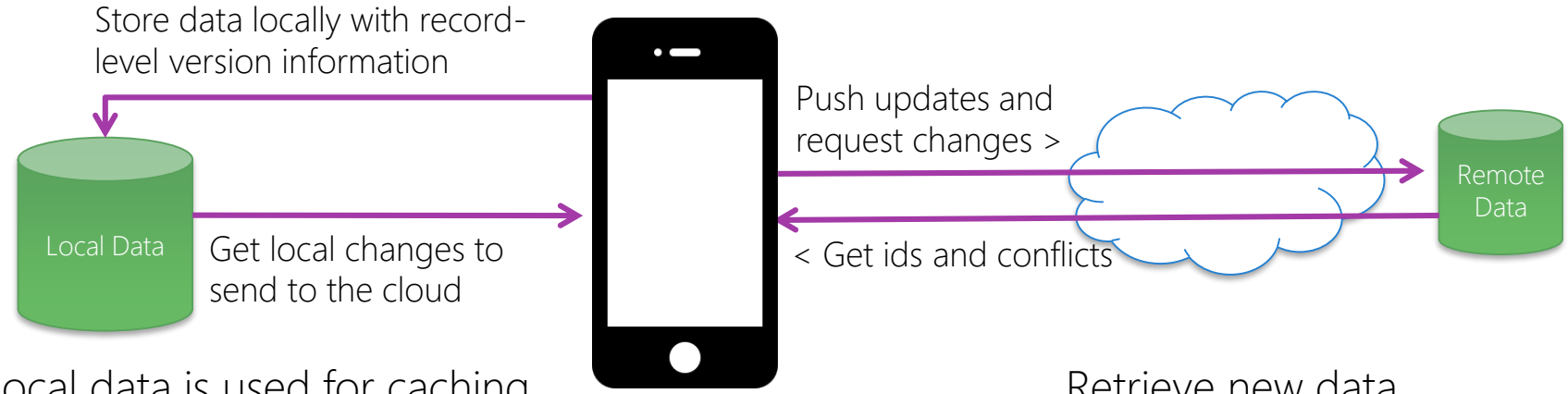
- Most networked application will use either a server based approach to data, a local cache of the data or the ability to work offline with both



Offline editing and synchronization



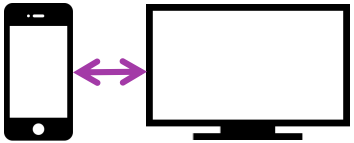
- Most networked application will use either a server based approach to data, a local cache of the data or the ability to work offline with both



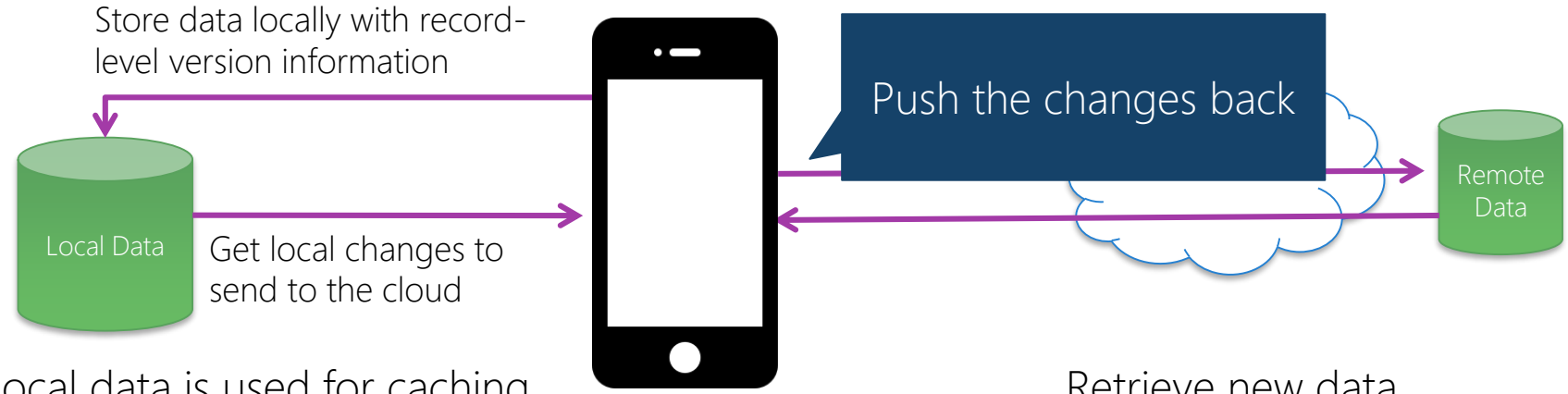
Local data is used for caching data and syncing local edits to a server

Retrieve new data and update server

Offline editing and synchronization



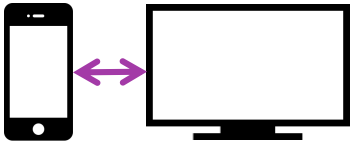
- Most networked application will use either a server based approach to data, a local cache of the data or the ability to work offline with both



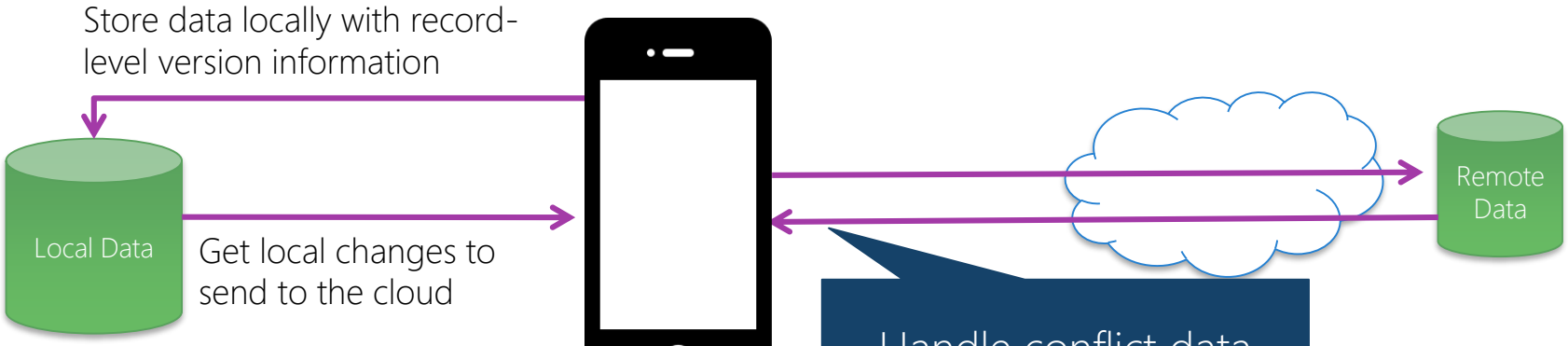
Local data is used for caching data and syncing local edits to a server

Retrieve new data and update server

Offline editing and synchronization



- Most networked application will use either a server based approach to data, a local cache of the data or the ability to work offline with both



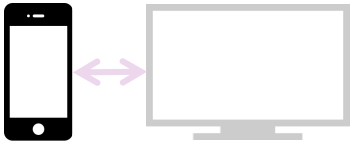
Local data is used for caching data and syncing local edits to a server

Handle conflict data and present options to the user

data
erver

Managing local data synching & conflict resolution significantly increases app complexity.

Creating objects to sync

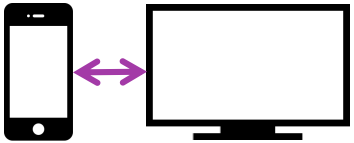


- Each business object should have its own entity - the class should represent the core information that you want to communicate between the server and the client

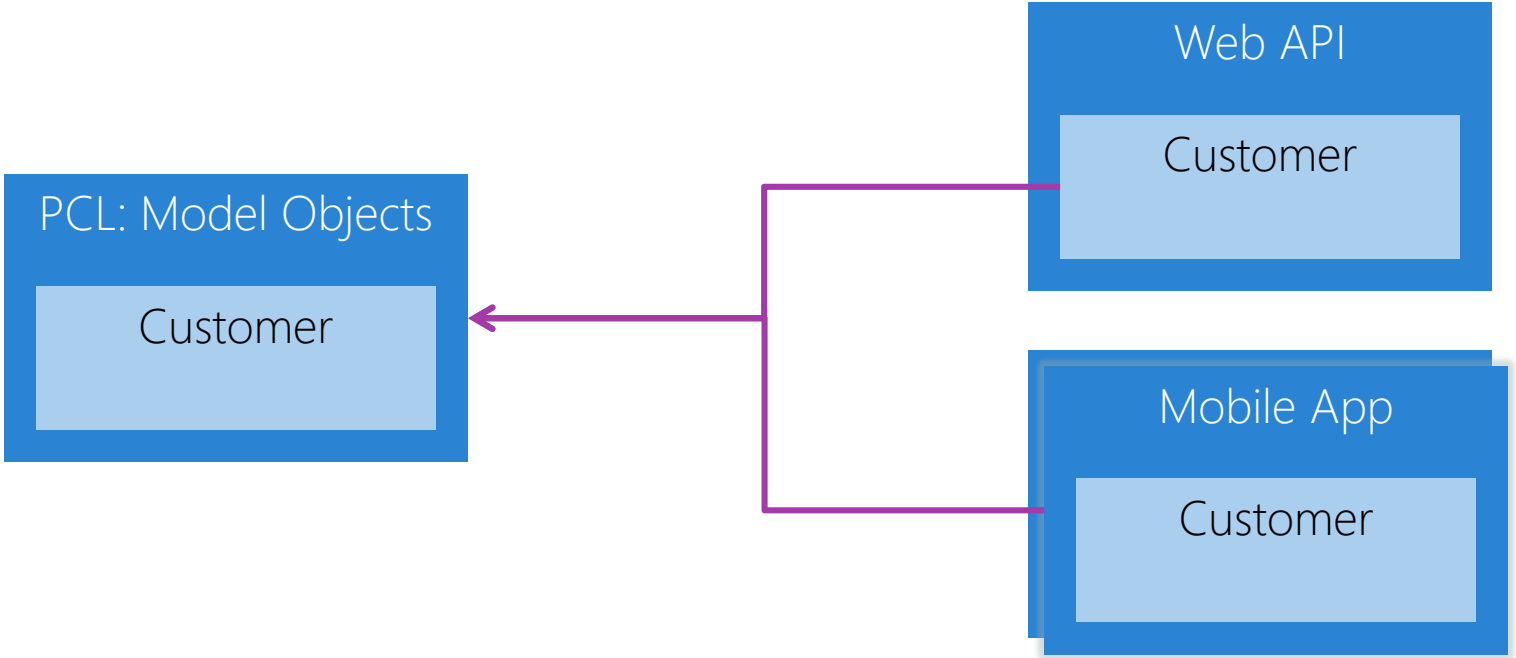
```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string Phone { get; set; }
    public string Notes { get; set; }
    public string[] Addresses { get; set; }
}
```

Define these classes in a separate PCL so they can be shared between the mobile app and Web Server

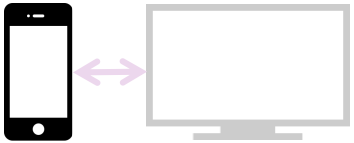
Reuse model classes



- Reuse model assemblies between the server and mobile clients to assist serialization and deserialization



Challenge: Inserting offline records



- When the remote server is not available you'll need to store a local primary key so when the server is eventually inserted you can map the generated primary to the local copy

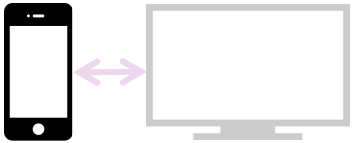
```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    ...

    public string CorrelationId { get; set; }
}
```

Id is the primary key on the server

CorrelationId is used for offline inserts

Challenge: Updating records



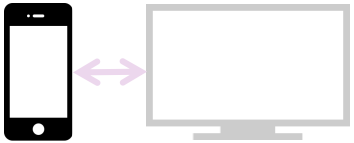
- When two devices have a local copy of the record and one applies an update to the record and the other device attempts to update the server, the second will be updating from a previous version

```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    ...

    public int VersionNumber { get; set; }
}
```

Version number is used to ensure you update the version of record you have been editing or report a conflict

Challenge: Deleting records



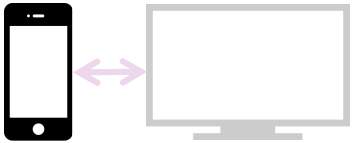
- When deleting records you should store the fact that a local record has been deleted, so synchronization will remove the record on the server

```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    ...

    public int VersionNumber { get; set; }
    public bool IsDeleted { get; set; }
}
```

The local deleted record and version number is used to tell the server what to remove when syncing

Challenge: Handling conflicts



- When update or delete conflicts occur you'll need to display the information about the records in conflict and also the timing details

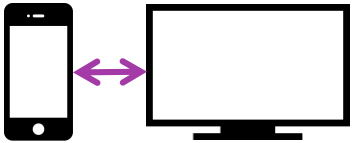
```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    ...

    public DateTime CreateDateTime { get; set; }
    public DateTime LastUpdateDateTime { get; set; }
    public DateTime DeletedDateTime { get; set; }
}
```

This information is used to display timing information about the conflict

Hint: use UTC for your records, so the server matches the clients

Structure your sync



- All this information is common to objects that will be synchronized, so it can be put in a base class

```
public class SyncObject
{
    public int Id { get; set; }
    public int VersionNumber { get; set; }

    public DateTime CreateDateTime { get; set; }
    public DateTime LastUpdateDateTime { get; set; }
    public DateTime DeletedDateTime { get; set; }

    public bool IsDeleted { get; set; }
    public string  CorrelationId { get; set; }
}
```

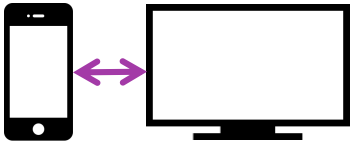
Use as a base class

Update only the version you edit

Store dates for later conflict display

CorrelationId is used for offline inserts

Structure your sync

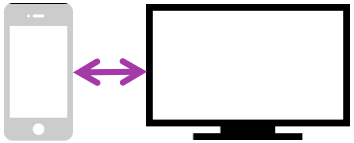


- The customer then becomes an extension of a class with synchronization capability

```
public class Customer : SyncObject
{
    public string Name { get; set; }
    public string Email { get; set; }
    public string Phone { get; set; }
    public string Notes { get; set; }
    public string[] Addresses { get; set; }
}
```

Entity objects become simpler and focus only on their data requirements

Processing sync updates

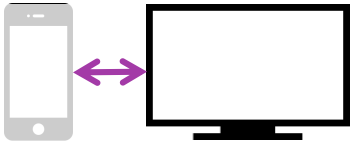


- On the server side, it is helpful to have an abstract capability to receive and process updates

```
public class BaseServerSync<T> where T : SyncObject
{
    public virtual T GetItem (T item);
    public virtual int Insert (T item);
    public virtual void Update (T item);
    public virtual void Delete (T item);
    ...
}
```

Need the ability to receive and process single items against a data store

Processing sync updates

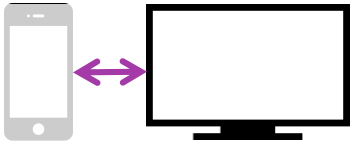


- On the server side, it is helpful to have an abstract capability to receive and process updates

```
public class BaseServerSync<T> where T : SyncObject
{
    ...
    public virtual void Audit (AuditAction action, T item);
    protected virtual void Setup ();
    protected virtual void Commit ();
    protected virtual void Rollback ();

    public virtual SyncResult<T> Process (IEnumerable<T> items,
        bool forceChanges = false);
}
```

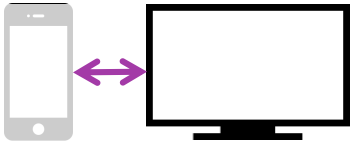
Processing sync updates



- Implement a custom subclass of sync to **`BaseServerSync`** to update an **`SyncObject`** so that changes can be applied

```
public class CustomerDataSync : BaseServerSync<Customer>
{
   ...
}
```

Processing sync updates



- Can use a WebApi method to receive changes from a client

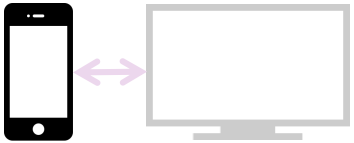
```
public class CustomersController : ApiController
{
    public CustomerDataSync _sync = new CustomerDataSync();

    public IEnumerable<Customer> Get() {
        return _sync.GetCustomers();
    }

    public SyncResult<Customer> Post([FromBody] Customer[] customers) {
        return _sync.Process(customers);
    }
    ...
```

WebApi is not a required mechanism, but it is a good mechanism for sharing the model code between the server and the mobile clients

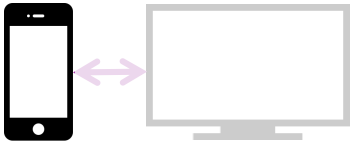
Connecting from the local data



- Send changes to the server and receive sync details

```
public async Task<SyncResult<Customer>> SyncData(List<Customer> items,
    bool forceChanges = false)
{
    using (var client = CreateRestClient())
    {
        string postBody = await JsonConvert.SerializeObjectAsync(items.ToArray());
        HttpResponseMessage getDataResponse;
        if (!forceChanges) {
            getDataResponse = await client.PostAsync ("",
                new StringContent (postBody, Encoding.UTF8, "application/json"));
        } else {
            getDataResponse = await client.PutAsync ("",
                new StringContent (postBody, Encoding.UTF8, "application/json"));
        }
```

Connecting from the local data



- Send receive changes between the server and the client.

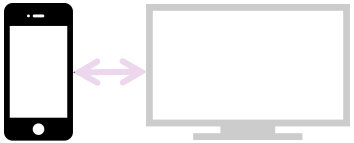
```
if (!getDataResponse.IsSuccessStatusCode)
                throw new CouldNotConnectException ();

        // Retrieve the JSON response
        jsonResponse = await getDataResponse.Content.ReadAsStringAsync()
            .ConfigureAwait(false);
    }

if (string.IsNullOrEmpty(jsonResponse))
        return null;

return await Task.Factory.StartNew(() =>
    JsonConvert.DeserializeObject<SyncResult<Customer>>(jsonResponse))
        .ConfigureAwait(false);
```

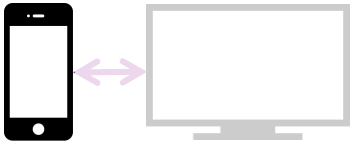
Platform-specific syncing



- For iOS 7 and above use Background Fetch mode to sync changes, even while the app is not running, can also use long running tasks

```
public override void PerformFetch (UIApplication application,
    Action<UIBackgroundFetchResult> completionHandler)
{
    try {
        var hasMoreData = await PerformSync();
        completionHandler(hasMoreData ? UIBackgroundFetchResult.NewData
             : UIBackgroundFetchResult.NoData);
    } catch {
         completionHandler(UIBackgroundFetchResult.Failed);
    }
}
```

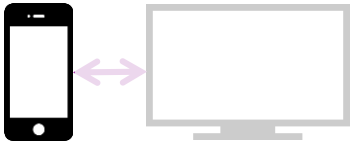
Platform-specific syncing



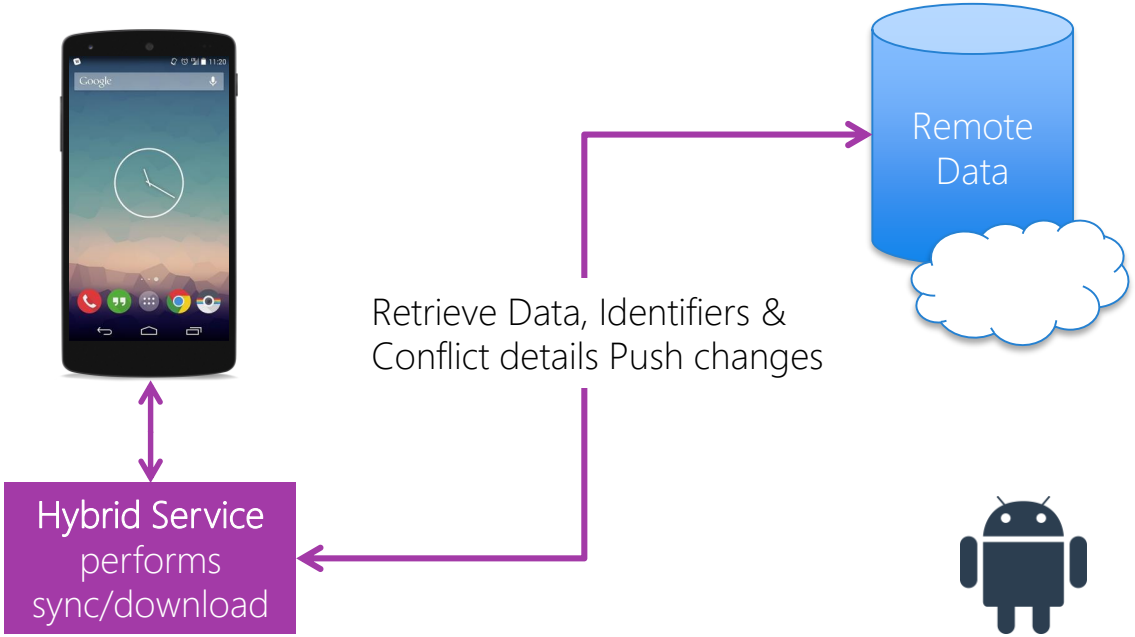
- For Android, ensure that synching is done via a Background service as the operations may take some time.

Hybrid Service performs sync/download

Platform-specific syncing

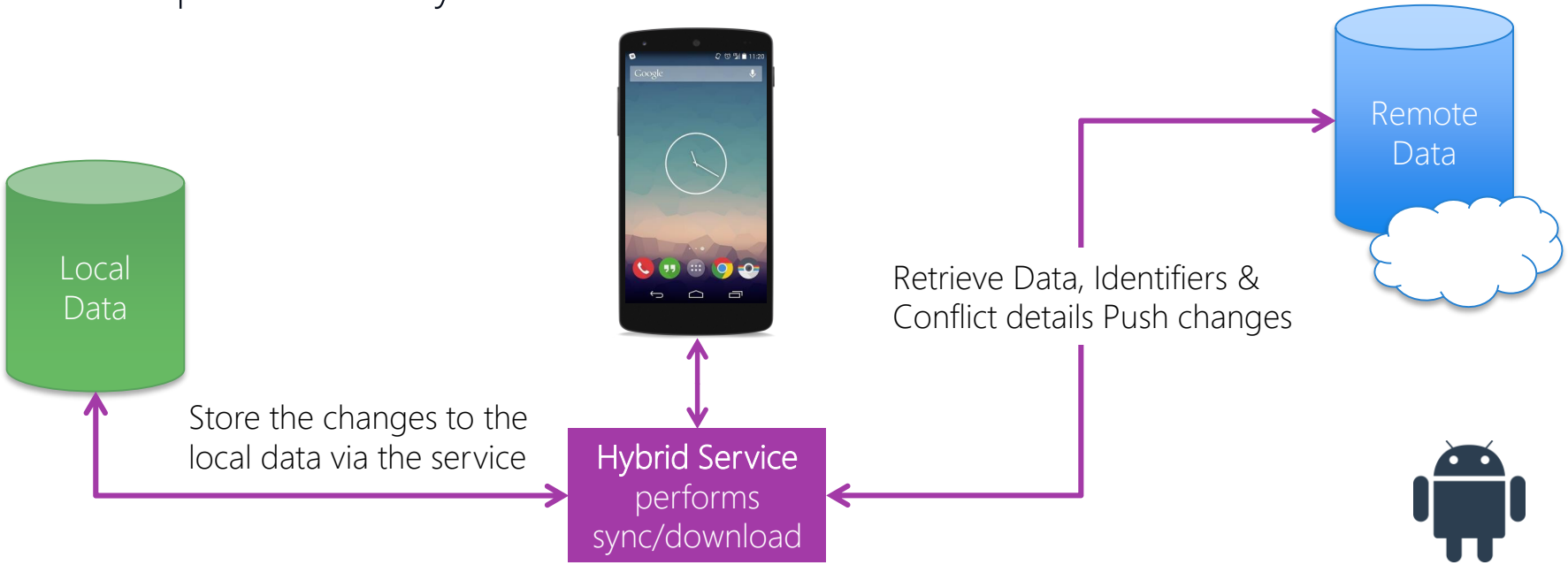


- For Android, ensure that synching is done via a Background service as the operations may take some time.

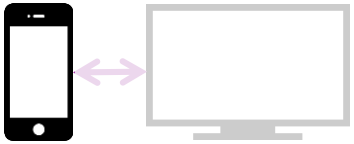


Platform-specific syncing

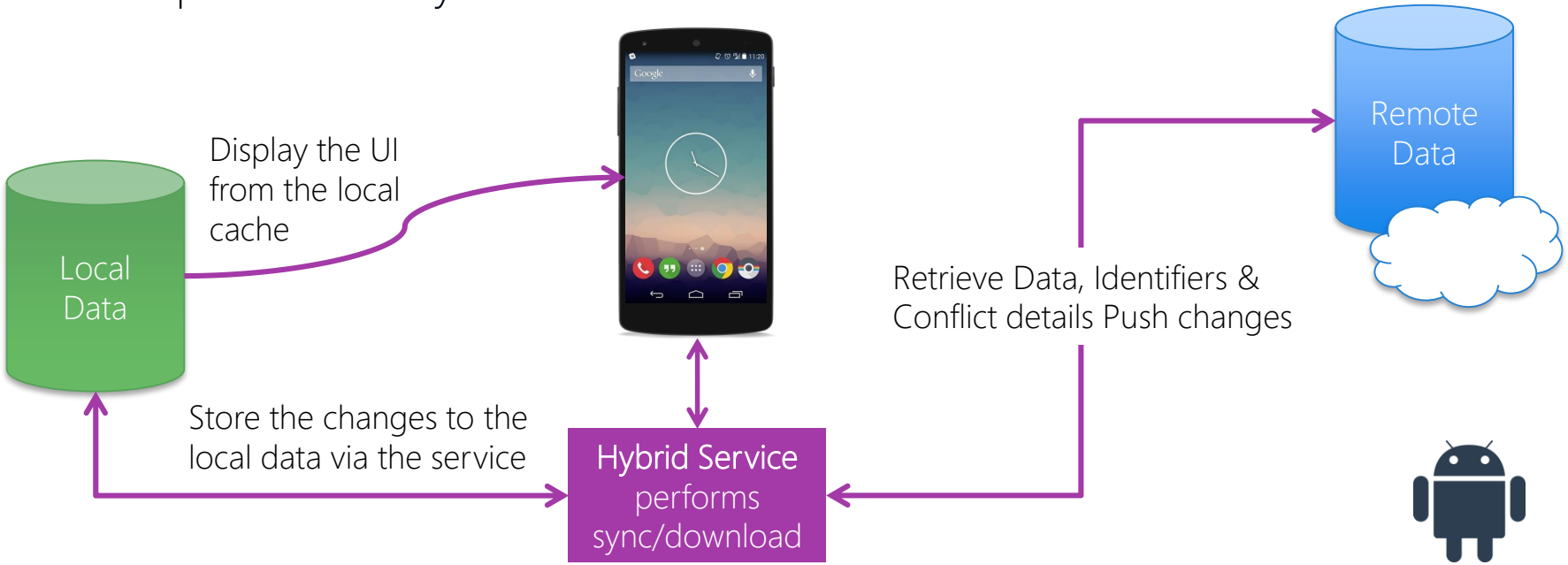
- For Android, ensure that synching is done via a Background service as the operations may take some time.



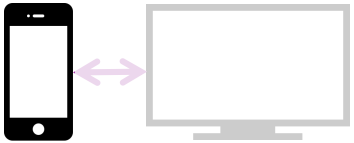
Platform-specific syncing



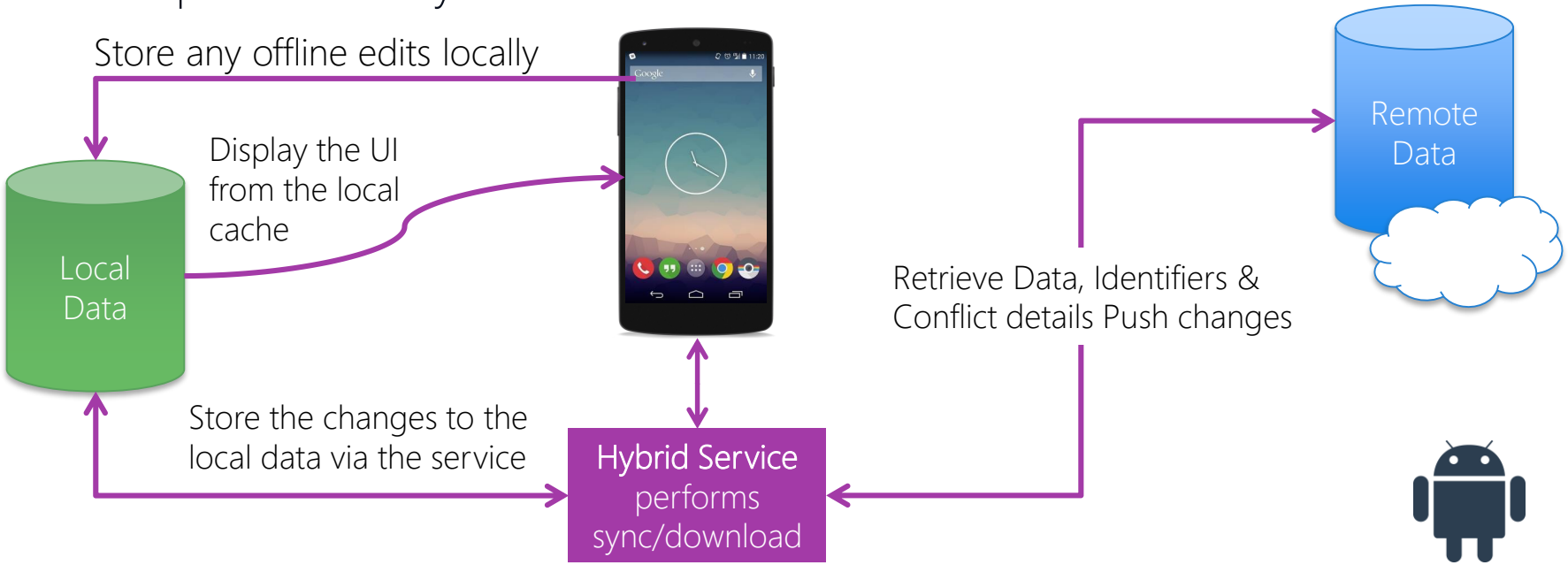
- For Android, ensure that synching is done via a Background service as the operations may take some time.



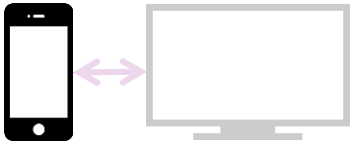
Platform-specific syncing



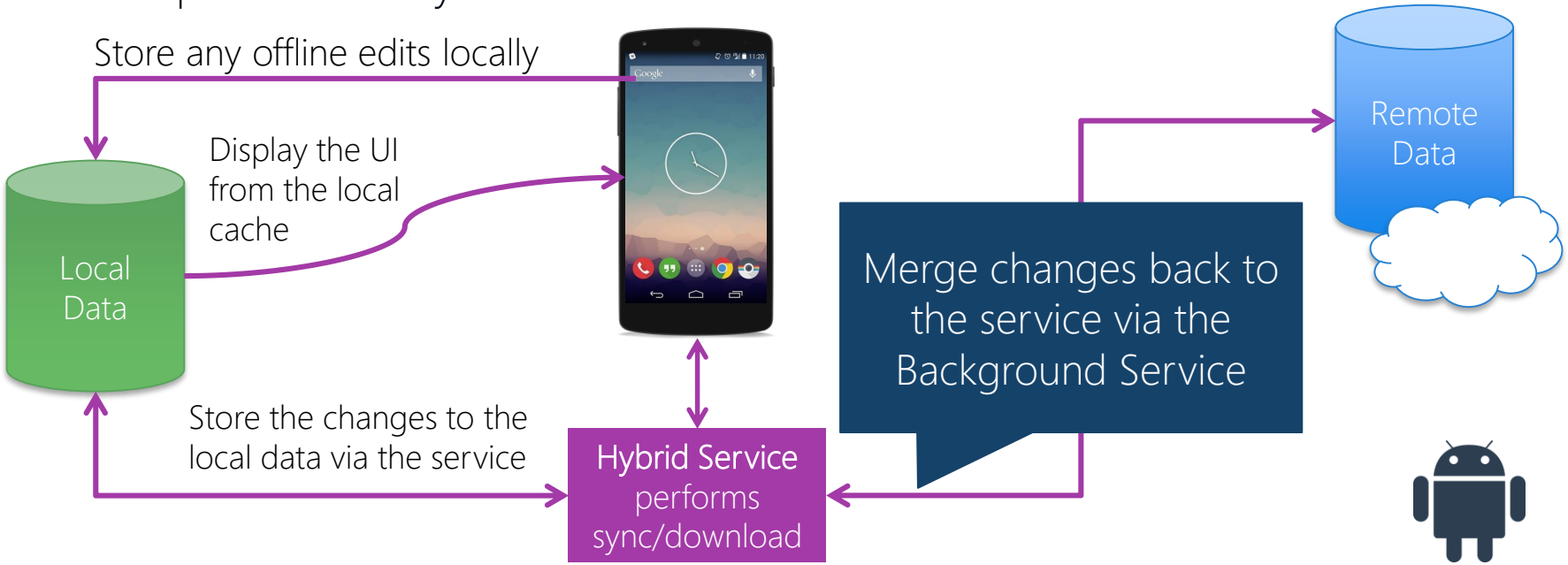
- For Android, ensure that synching is done via a Background service as the operations may take some time.



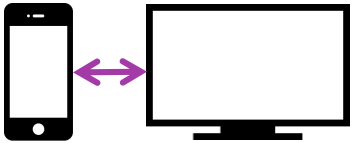
Platform-specific syncing



- For Android, ensure that synching is done via a Background service as the operations may take some time.



Handling conflicts



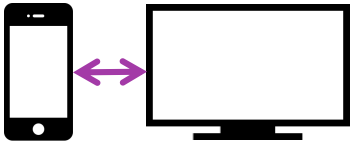
- There are two core conflicts that can occur, where users are updating the same record or where a user is updating a deleted record

Customer Version: 1

When updating/deleting the same record what happens?

Customer Version: 1

Handling update conflicts



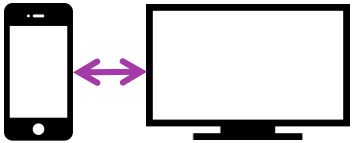
- When multiple users are editing the same record, who/what decides on which edit should be made

Customer Version: 1

Where one record has changed, bring down both records and let the user decide and audit the result

Customer Version: 2

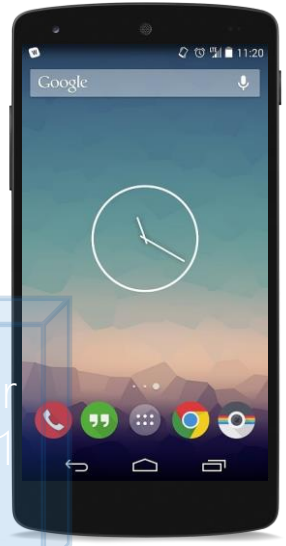
Handling delete conflicts



- When a user is updating a deleted record, what happens?

Customer
Version: 1

Inform the user that the delete has happened and encourage them to resynchronize their data



Customer
Version: 1
Deleted

Demonstration

A coded offline synchronization mechanism

Xamarin University

Flash Quiz

Xamarin University

Flash Quiz

① What are some of the challenges of synchronizing local offline data remotely (Choose all that apply)

a) Lack of network connectivity

b) No local stored procedures

c) Creating identifiers to insert

d) Handling conflicts between devices updating

e) All of the above

Flash Quiz

① What are some of the challenges of synchronizing local offline data remotely (Choose all that apply)

a) Lack of network connectivity
b) No local stored procedures
c) Creating identifiers to insert
d) Handling conflicts between devices updating
e) All of the above

Flash Quiz

② If you have offline editing capability, you don't need to warn users that there is no connection

a) True

b) False

Flash Quiz

② If you have offline editing capability, you don't need to warn users that there is no connection

a) True

b) False

Flash Quiz

③ When handling conflicts between two devices/users who have updated the record, you should:

a) Make the last update win

b) Display the two results and let the user decide

c) It depends on the business rules of the synchronization system

d) Make sure the records version numbers are the same

e) Reject the update

f) None of the above

g) Answers a)-d)

Flash Quiz

③ When handling conflicts between two devices/users who have updated the record, you should:

a) Make the last update win

b) Display the two results and let the user decide

c) <u>It depends on the business rules of the synchronization system</u>

d) Make sure the records version numbers are the same

e) Reject the update

f) None of the above

g) Answers a)-d)

Tasks

1. Discuss and establish possible patterns for offline editing synchronization
2. Examine challenges with Synchronization

Evaluate Data Sync Tools

Tasks

1. Evaluate third-party options

Data sync code cost

- There is a significant amount of time writing, testing, deploying and managing a synchronized mechanism.

Time to develop a sync capability maybe too large

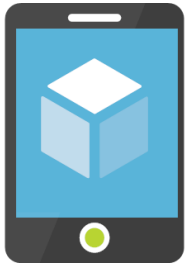
The risk of new development maybe too high

Enterprises may be affected by opportunity cost losses by being late to market

Coding your own sync gives you significantly more control.

Making pragmatic choices

- Writing Synchronization code is difficult and complex. If you can find a third-party offering you may complete your project sooner



Azure Mobile Apps

Couchbase

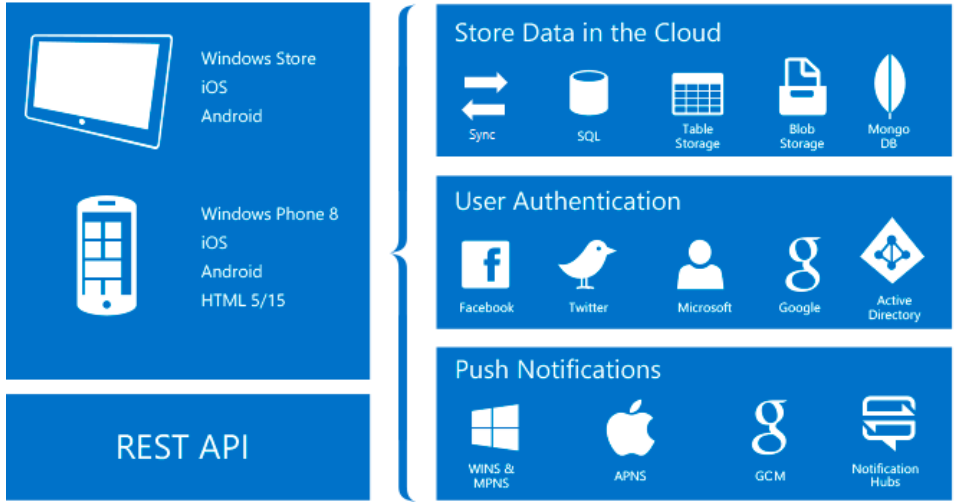
Zumero

You should always evaluate tools for suitability in *your* specific project

Azure Mobile Apps

- Azure Mobile Apps are a component of Azure App Service, which is a highly scalable mobile development platform that greatly simplify data caching and synchronization

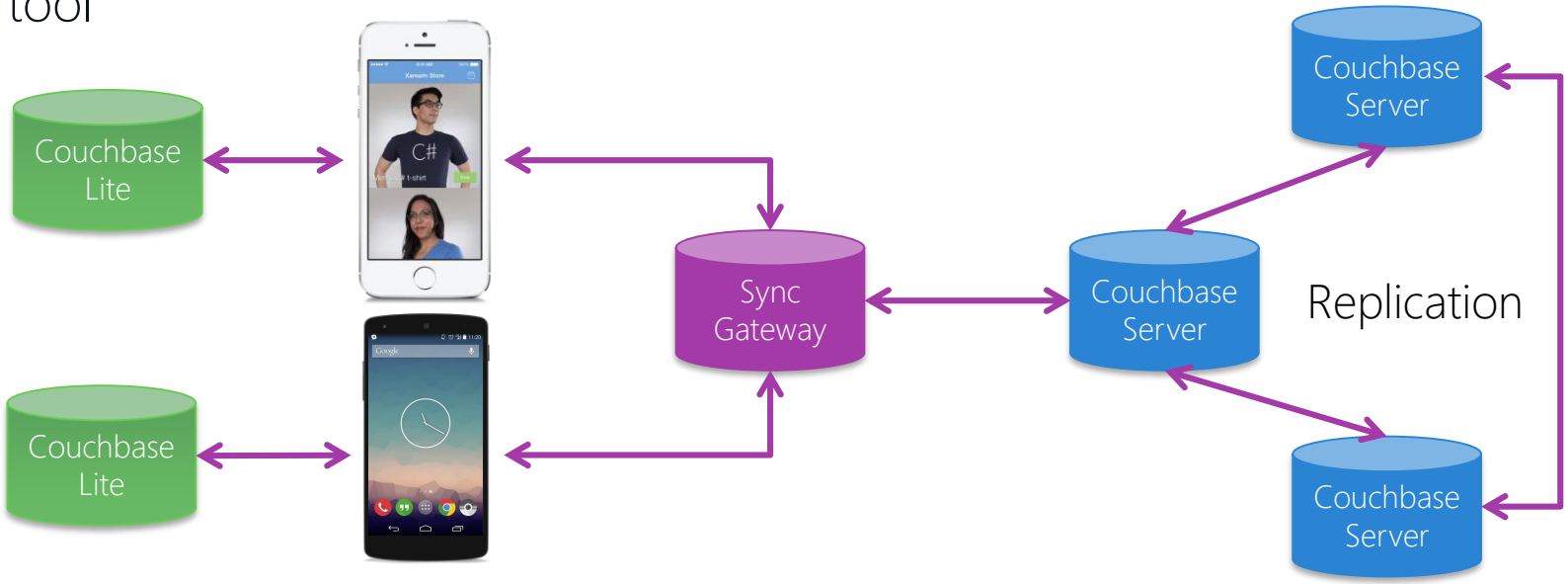
Broad mobile platform support including Windows, iOS, Android and Xamarin.Forms



For more information – checkout the Azure classes in Xamarin University

Couchbase

- Couchbase is a NoSQL-based data that by defaults supports replication between systems, including mobile devices through their SyncGateway tool



Couchbase

- Has very powerful replication support between the systems which almost feels like real-time communication

Pros	Cons
▪ Very quick/powerful replication ▪ LiveUpdates can be hooked into ViewModels which makes it good for Xamarin.Forms	▪ Requires a long setup process to begin working ▪ Requires understanding of NoSQL techniques such as map reduce ▪ Works best for Greenfields projects

A high-quality example of a Couchbase Xamarin.Forms app with Data Synchronization can be found at https://github.com/FireflyLogic/couchbase-connect-14

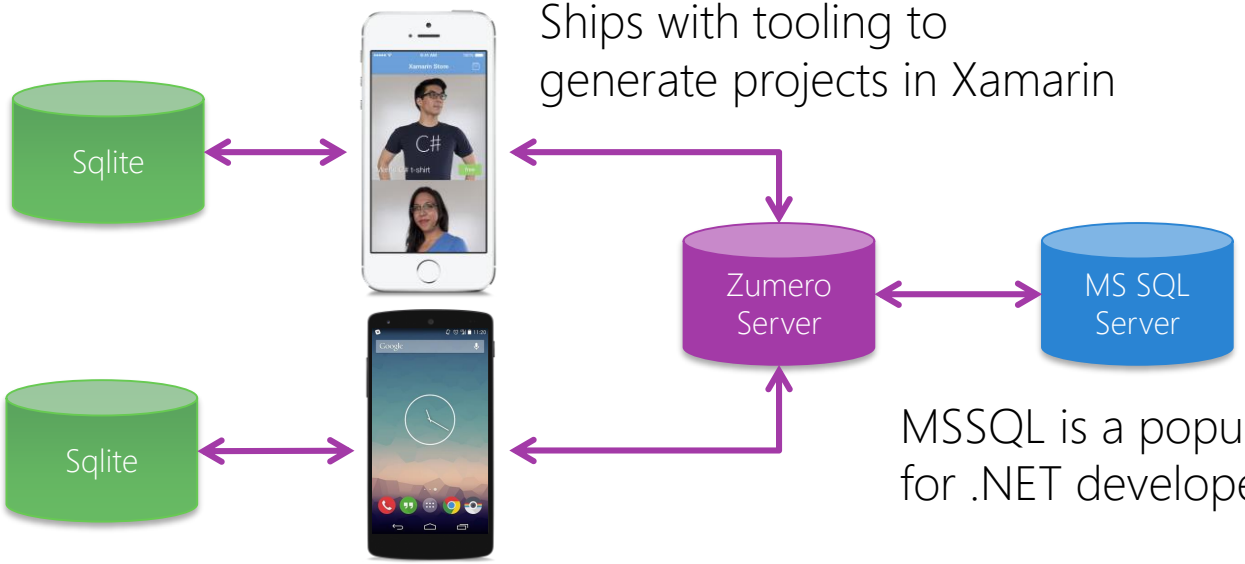
Demonstration

Couchbase synchronization

Xamarin University

Zumero

- SQL based Synching between a mobile application and a customer system



Ships with tooling to generate projects in Xamarin

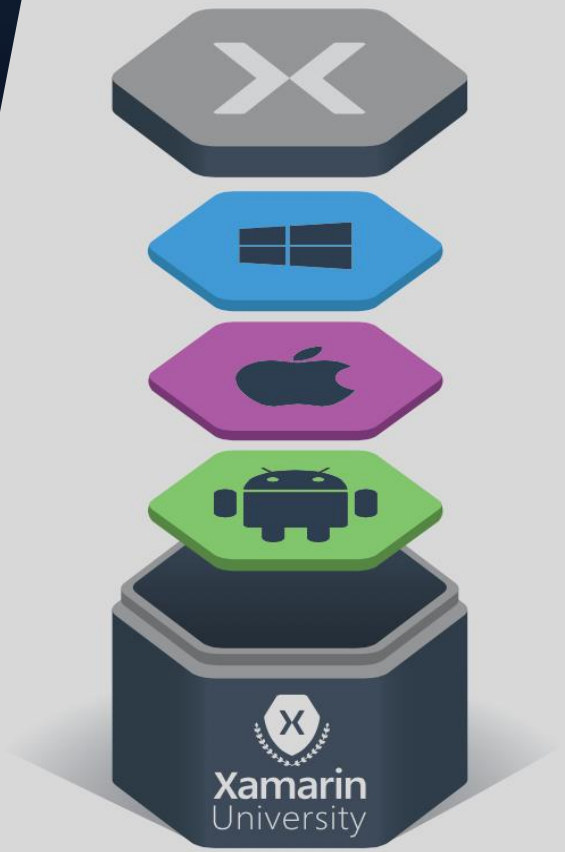
MSSQL is a popular backend for .NET developers

Summary

1. Evaluate third-party options

Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile



Microsoft