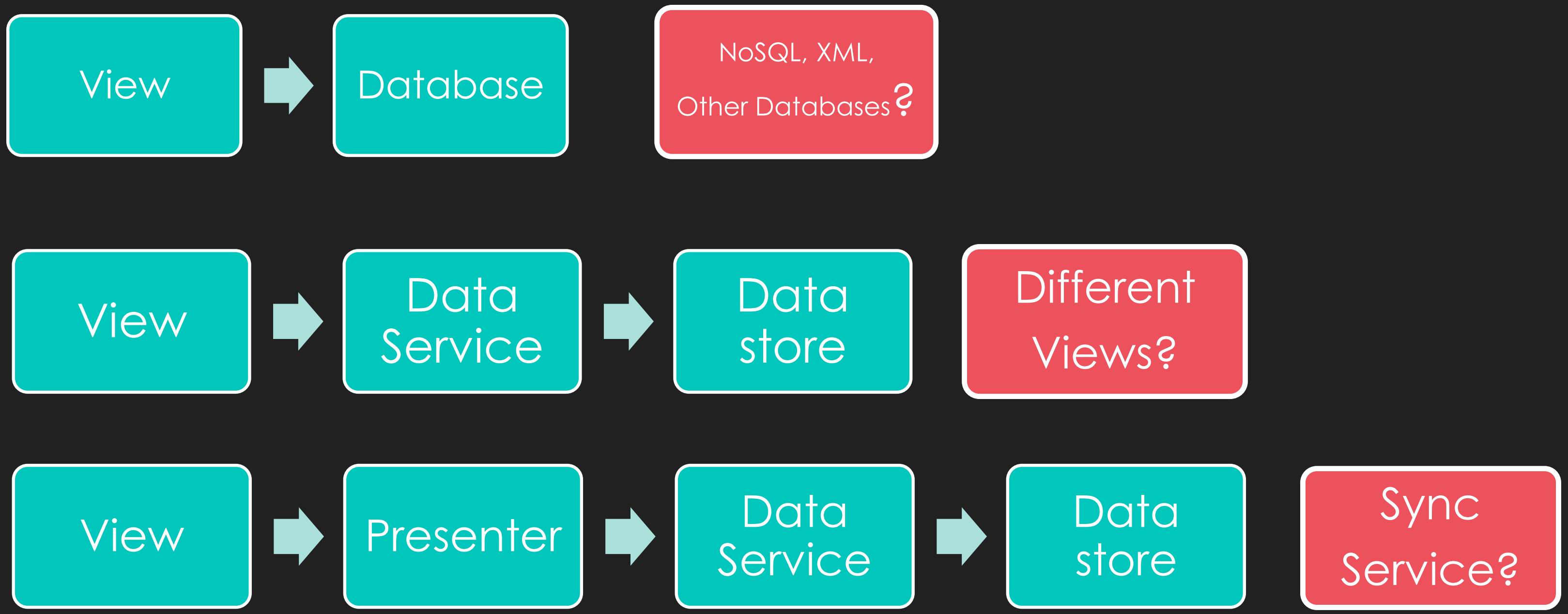
Note App

Rajasekar Pandiyan

Requirements

- Windows client to create notebooks that contain notes
- Notes have title and text content
- Notes gets saved in cloud
- Notes can also be saved using Web application
- Windows client should work in offline mode
- Switching back to online mode should sync the local changes to the server and vice versa
- Conflicting changes are resolved by preferring local changes over the modification done by web application

Considerations and approaches



1. Two layer approach

View → Database

NoSQL, XML, Other Databases?

2. Database abstraction

View → Data Service → Data store

Different Views?

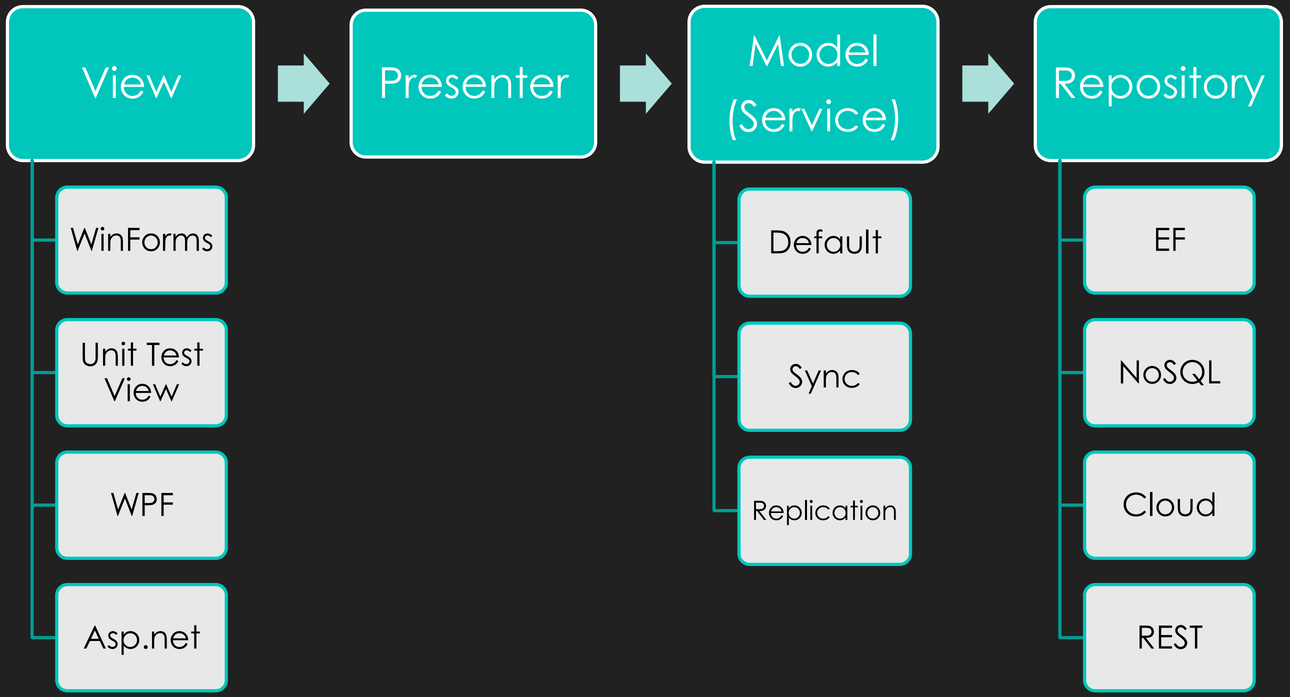
3. View abstraction

View → Presenter → Data Service → Data store

Sync Service?

Final approach

Four layer architecture

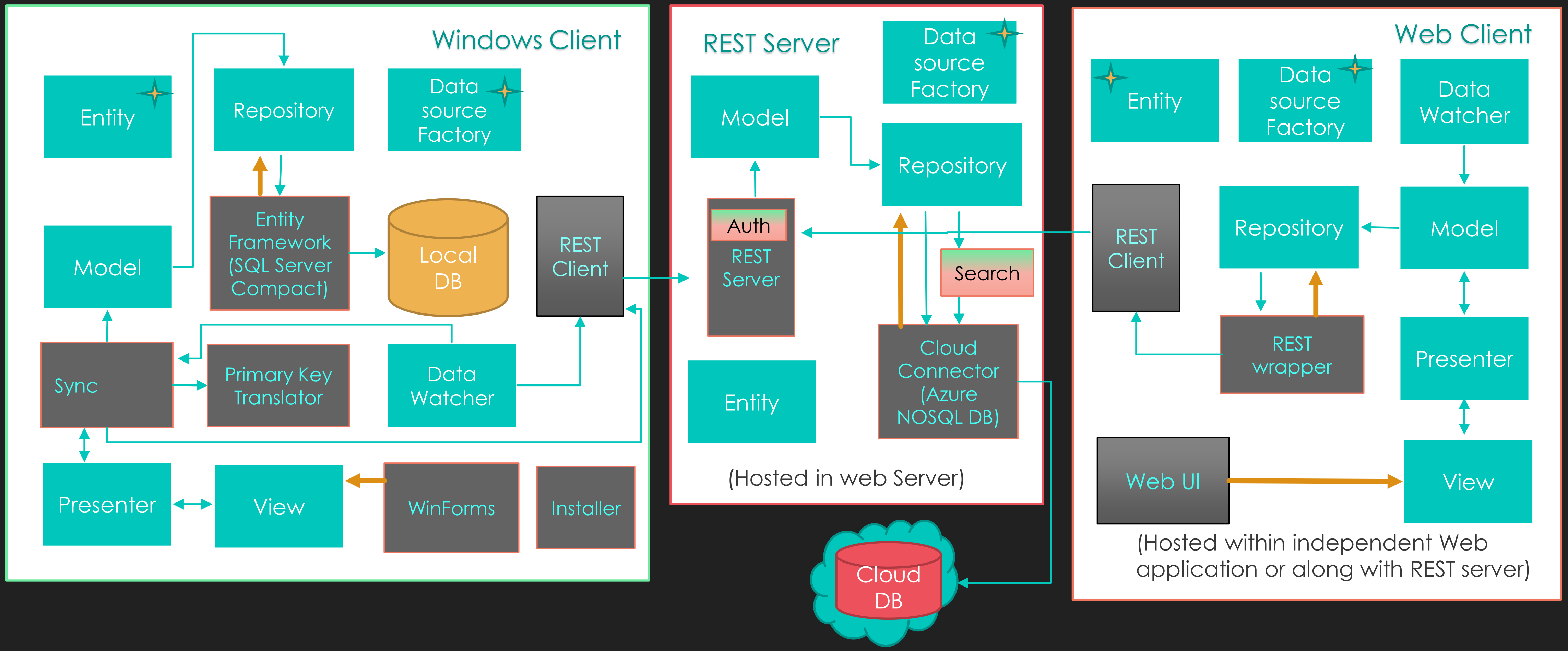


Multi layer design

- All the layers are abstract types
- The dependency for each layer composed using abstract type (Interface)
- The dependency are injected via constructor
- Constructor DI enables all components for UT
- Every layer is independent of each other
- Concrete classes are instantiated and passed to the constructor using bootstrap module

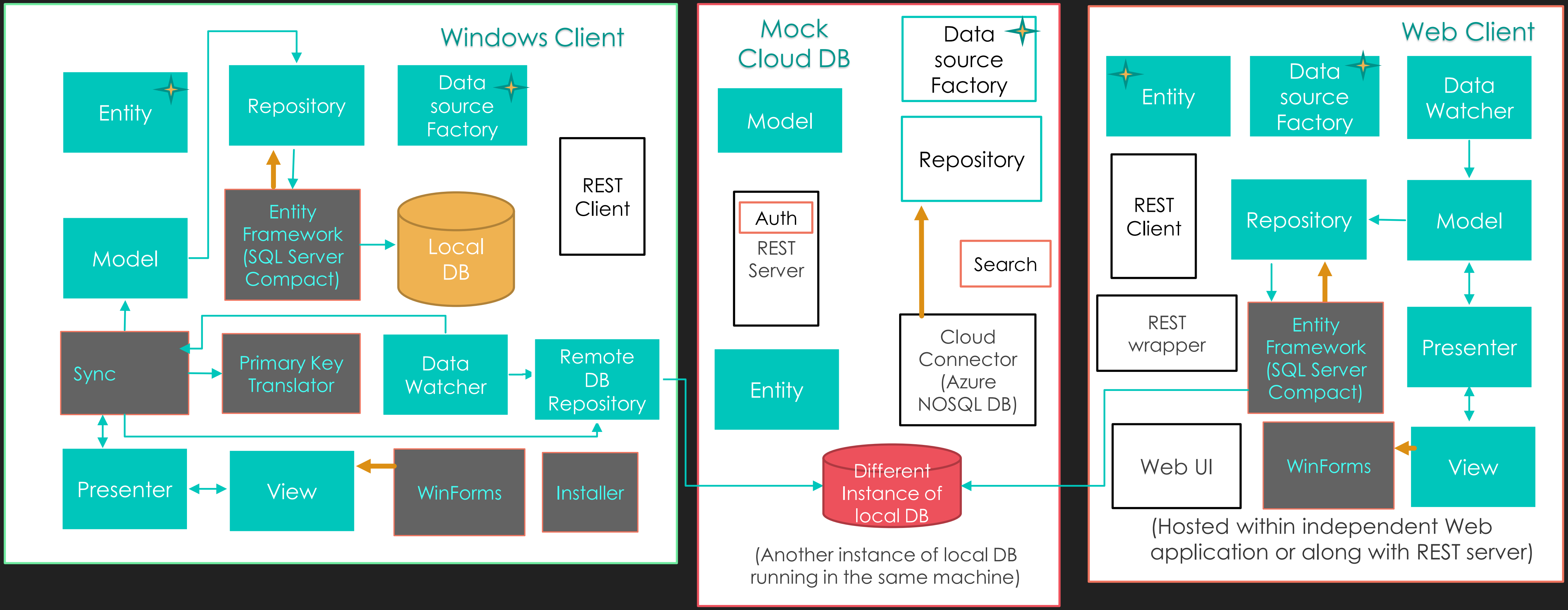
Overall architecture

- 🟩 Abstract components
- ⬛ Deployment specific implementation components
- ✦ Helper or cross cutting components
- ← Implementation from abstract components



Demo deployment

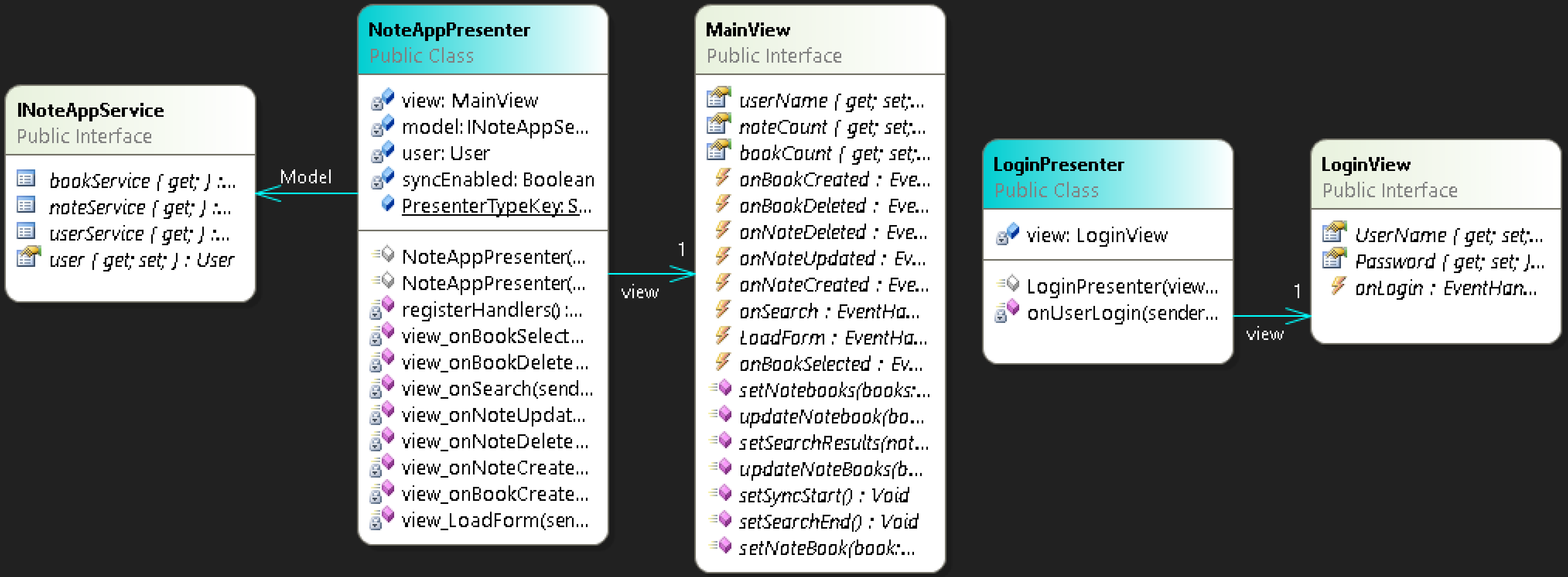
- ■ Common components
- ■ Deployment specific Components
- ✦ Helper or cross cutting components
- □ Scoped out for demo
- ← Implementation from abstract components



Projects

- Core – Defines all the core interfaces and default implementation for abstract types for Model and Persistence layer
- MVP – Defines presentation layer and View interfaces
- UI – Implements view interface defined in MVP project
- Database – Implements IRepository defined in Core with EF
- Bootstrap – Builds complex deployment recipes
- Installer
- Test(UT) – Defines test cases for NoteAppService and SyncNoteAppService
- Console Test – Simple manual verification of db data
- Sync – Implements INoteAppService defined in Core to provide Sync functionality

NoteMVP



INoteAppService
Public Interface

- bookService { get; } :...
- noteService { get; } :...
- userService { get; } :...
- user { get; set; } : User

NoteAppPresenter
Public Class

- view: MainView
- model: INoteAppSe...
- user: User
- syncEnabled: Boolean
- PresenterTypeKey: S...

- NoteAppPresenter(...
- NoteAppPresenter(...
- registerHandlers() :...
- view_onBookSelect...
- view_onBookDelete...
- view_onSearch(send...
- view_onNoteUpdat...
- view_onNoteDelete...
- view_onNoteCreate...
- view_onBookCreate...
- view_LoadForm(sen...

MainView
Public Interface

- userName { get; set;...
- noteCount { get; set;...
- bookCount { get; set;...
- onBookCreated : Eve...
- onBookDeleted : Eve...
- onNoteDeleted : Eve...
- onNoteUpdated : Ev...
- onNoteCreated : Eve...
- onSearch : EventHa...
- LoadForm : EventHa...
- onBookSelected : Ev...
- setNotebooks(books:...
- updateNotebook(bo...
- setSearchResults(not...
- updateNoteBooks(b...
- setSyncStart() : Void
- setSearchEnd() : Void
- setNoteBook(book:...

LoginPresenter
Public Class

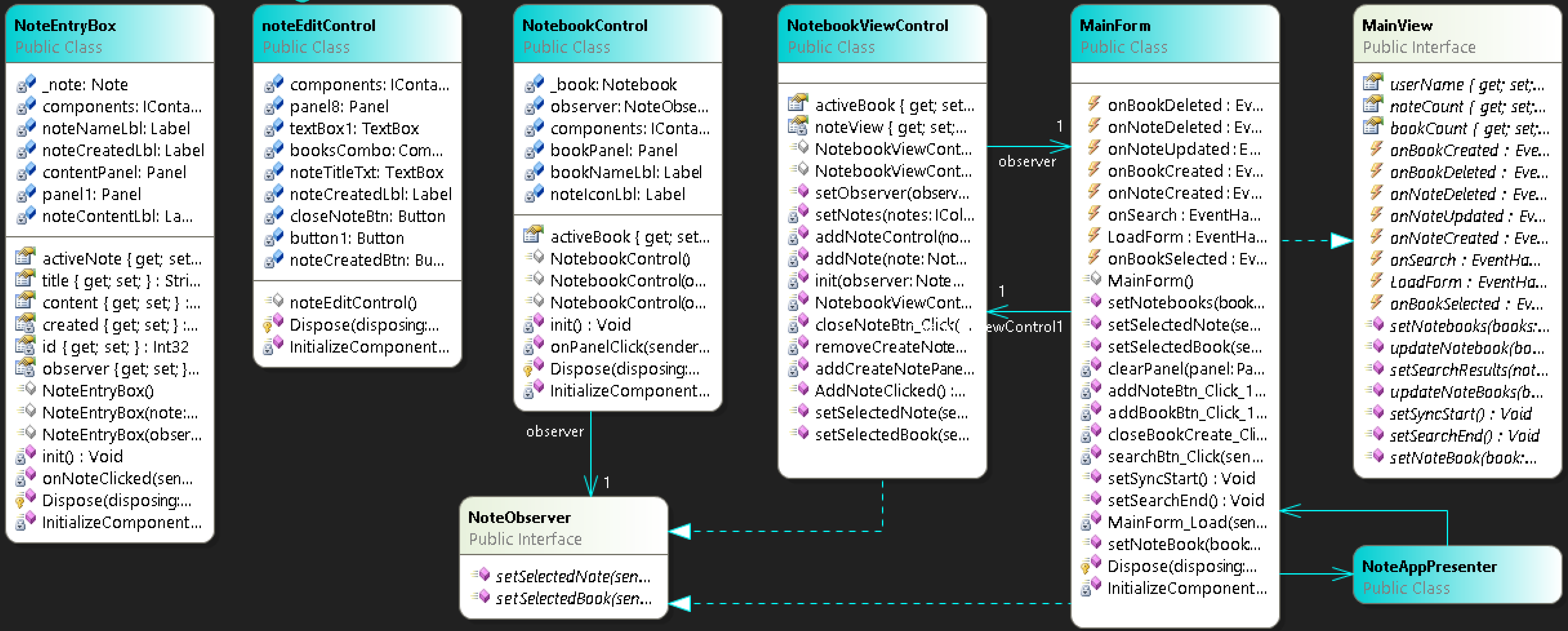
- view: LoginView

- LoginPresenter(view...
- onUserLogin(sender...

LoginView
Public Interface

- UserName { get; set;...
- Password { get; set; }...
- onLogin : EventHan...

Windows UI



NoteEntryBox
Public Class

- _note: Note
- components: IConta...
- noteNameLbl: Label
- noteCreatedLbl: Label
- contentPanel: Panel
- panel1: Panel
- noteContentLbl: La...

- activeNote { get; set...
- title { get; set; } : Stri...
- content { get; set; } :...
- created { get; set; } :...
- id { get; set; } : Int32
- observer { get; set; }...
- NoteEntryBox()
- NoteEntryBox(note:...
- NoteEntryBox(obser...
- init() : Void
- onNoteClicked(sen...
- Dispose(disposing:...
- InitializeComponent...

noteEditControl
Public Class

- components: IConta...
- panel8: Panel
- textBox1: TextBox
- booksCombo: Com...
- noteTitleTxt: TextBox
- noteCreatedLbl: Label
- closeNoteBtn: Button
- button1: Button
- noteCreatedBtn: Bu...

- noteEditControl()
- Dispose(disposing:...
- InitializeComponent...

NotebookControl
Public Class

- _book: Notebook
- observer: NoteObse...
- components: IConta...
- bookPanel: Panel
- bookNameLbl: Label
- noteIconLbl: Label

- activeBook { get; set...
- NotebookControl()
- NotebookControl(o...
- NotebookControl(o...
- init() : Void
- onPanelClick(sender...
- Dispose(disposing:...
- InitializeComponent...

NotebookViewControl
Public Class

- activeBook { get; set...
- noteView { get; set;...
- NotebookViewCont...
- NotebookViewCont...
- setObserver(observ...
- setNotes(notes: ICol...
- addNoteControl(no...
- addNote(note: Not...
- init(observer: Note...
- NotebookViewCont...
- closeNoteBtn_Click(.
- removeCreateNote...
- addCreateNotePane...
- AddNoteClicked() :...
- setSelectedNote(se...
- setSelectedBook(se...

MainForm
Public Class

- onBookDeleted : Ev...
- onNoteDeleted : Ev...
- onNoteUpdated : E...
- onBookCreated : Ev...
- onNoteCreated : Ev...
- onSearch : EventHa...
- LoadForm : EventHa...
- onBookSelected : Ev...
- MainForm()
- setNotebooks(book...
- setSelectedNote(se...
- setSelectedBook(se...
- clearPanel(panel: Pa...
- addNoteBtn_Click_1...
- addBookBtn_Click_1...
- closeBookCreate_Cli...
- searchBtn_Click(sen...
- setSyncStart() : Void
- setSearchEnd() : Void
- MainForm_Load(sen...
- setNoteBook(book...
- Dispose(disposing:...
- InitializeComponent...

MainView
Public Interface

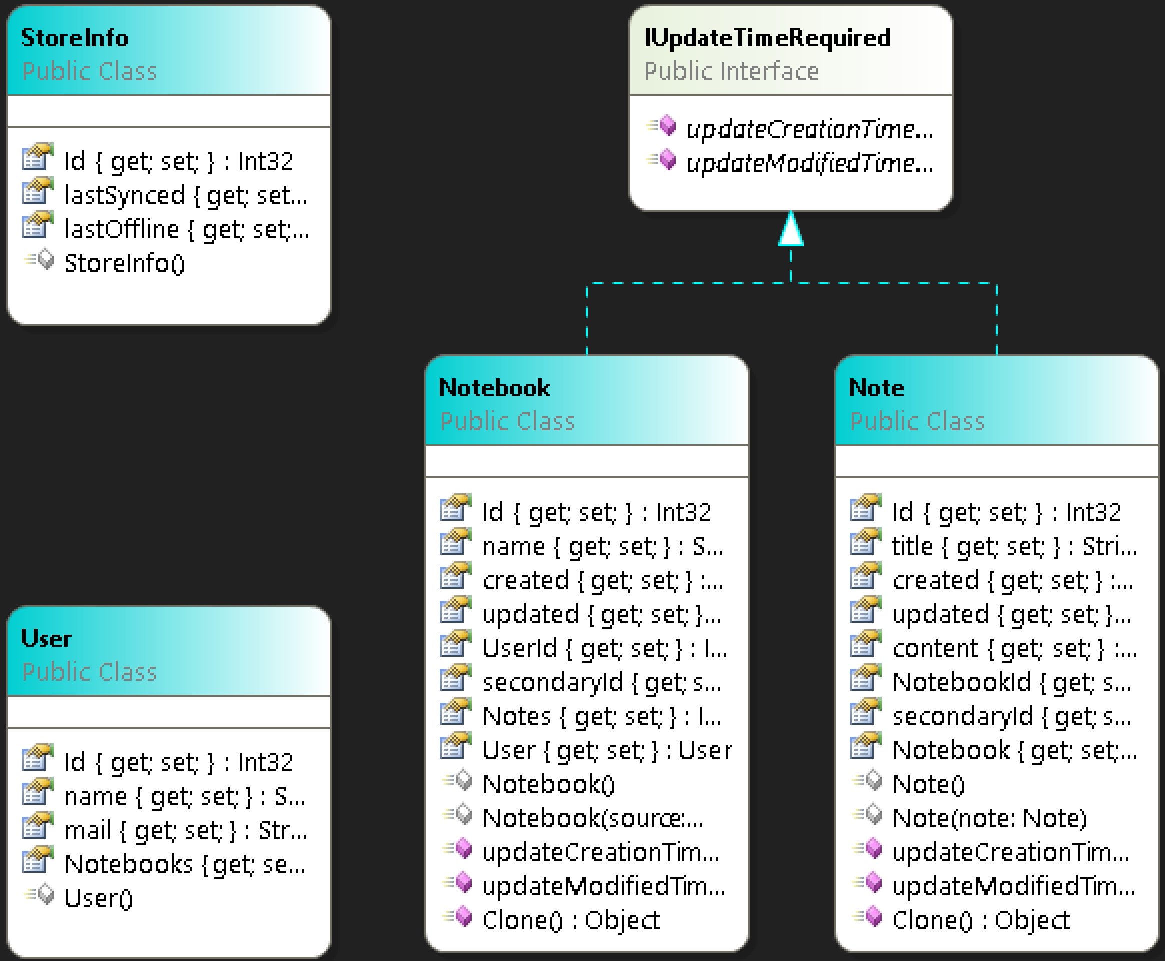
- *userName { get; set;...*
- *noteCount { get; set;...*
- *bookCount { get; set;...*
- *onBookCreated : Eve...*
- *onBookDeleted : Eve...*
- *onNoteDeleted : Eve...*
- *onNoteUpdated : Ev...*
- *onNoteCreated : Eve...*
- *onSearch : EventHa...*
- *LoadForm : EventHa...*
- *onBookSelected : Ev...*
- *setNotebooks(books:...*
- *updateNotebook(bo...*
- *setSearchResults(not...*
- *updateNoteBooks(b...*
- *setSyncStart() : Void*
- *setSearchEnd() : Void*
- *setNoteBook(book:...*

NoteObserver
Public Interface

- *setSelectedNote(sen...*
- *setSelectedBook(sen...*

NoteAppPresenter
Public Class

NoteAppCore(Entity)



StoreInfo
Public Class

Id { get; set; } : Int32
lastSynced { get; set...
lastOffline { get; set;...
StoreInfo()

IUpdateTimeRequired
Public Interface

updateCreationTime...
updateModifiedTime...

Notebook
Public Class

Id { get; set; } : Int32
name { get; set; } : S...
created { get; set; } :...
updated { get; set; }...
UserId { get; set; } : I...
secondaryId { get; s...
Notes { get; set; } : I...
User { get; set; } : User
Notebook()
Notebook(source:...
updateCreationTim...
updateModifiedTim...
Clone() : Object

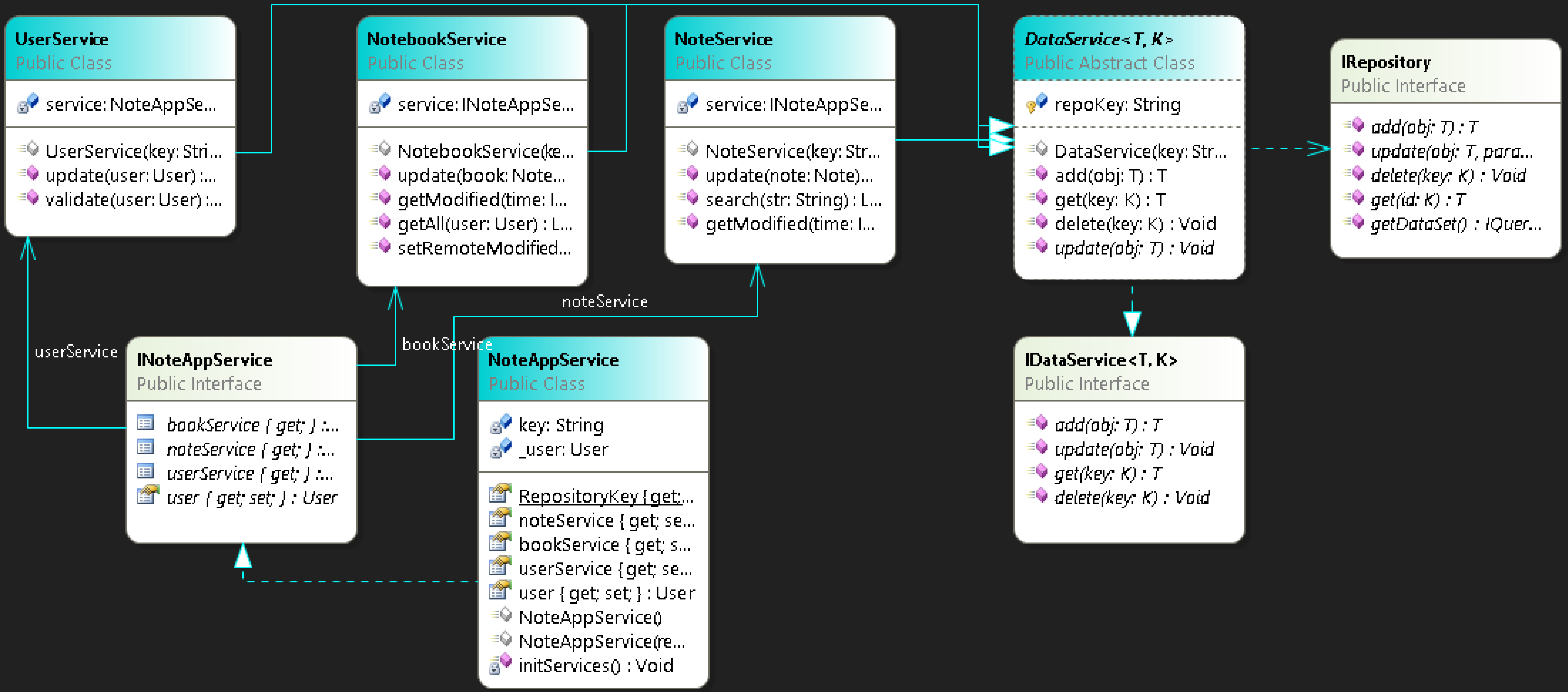
Note
Public Class

Id { get; set; } : Int32
title { get; set; } : Stri...
created { get; set; } :...
updated { get; set; }...
content { get; set; } :...
NotebookId { get; s...
secondaryId { get; s...
Notebook { get; set;...
Note()
Note(note: Note)
updateCreationTim...
updateModifiedTim...
Clone() : Object

User
Public Class

Id { get; set; } : Int32
name { get; set; } : S...
mail { get; set; } : Str...
Notebooks { get; se...
User()

NoteAppCore (Service)



UserService
Public Class

service: NoteAppSe...

UserService(key: Stri...
update(user: User) :...
validate(user: User) :...

NotebookService
Public Class

service: INoteAppSe...

NotebookService(ke...
update(book: Note...
getModified(time: I...
getAll(user: User) : L...
setRemoteModified...

NoteService
Public Class

service: INoteAppSe...

NoteService(key: Str...
update(note: Note)...
search(str: String) : L...
getModified(time: I...

DataService< T, K >
Public Abstract Class

repoKey: String

DataService(key: Str...
add(obj: T) : T
get(key: K) : T
delete(key: K) : Void
update(obj: T) : Void

IRepository
Public Interface

add(obj: T) : T
update(obj: T, para...
delete(key: K) : Void
get(id: K) : T
getDataSet() : IQuer...

INoteAppService
Public Interface

bookService { get; } :...
noteService { get; } :...
userService { get; } :...
user { get; set; } : User

NoteAppService
Public Class

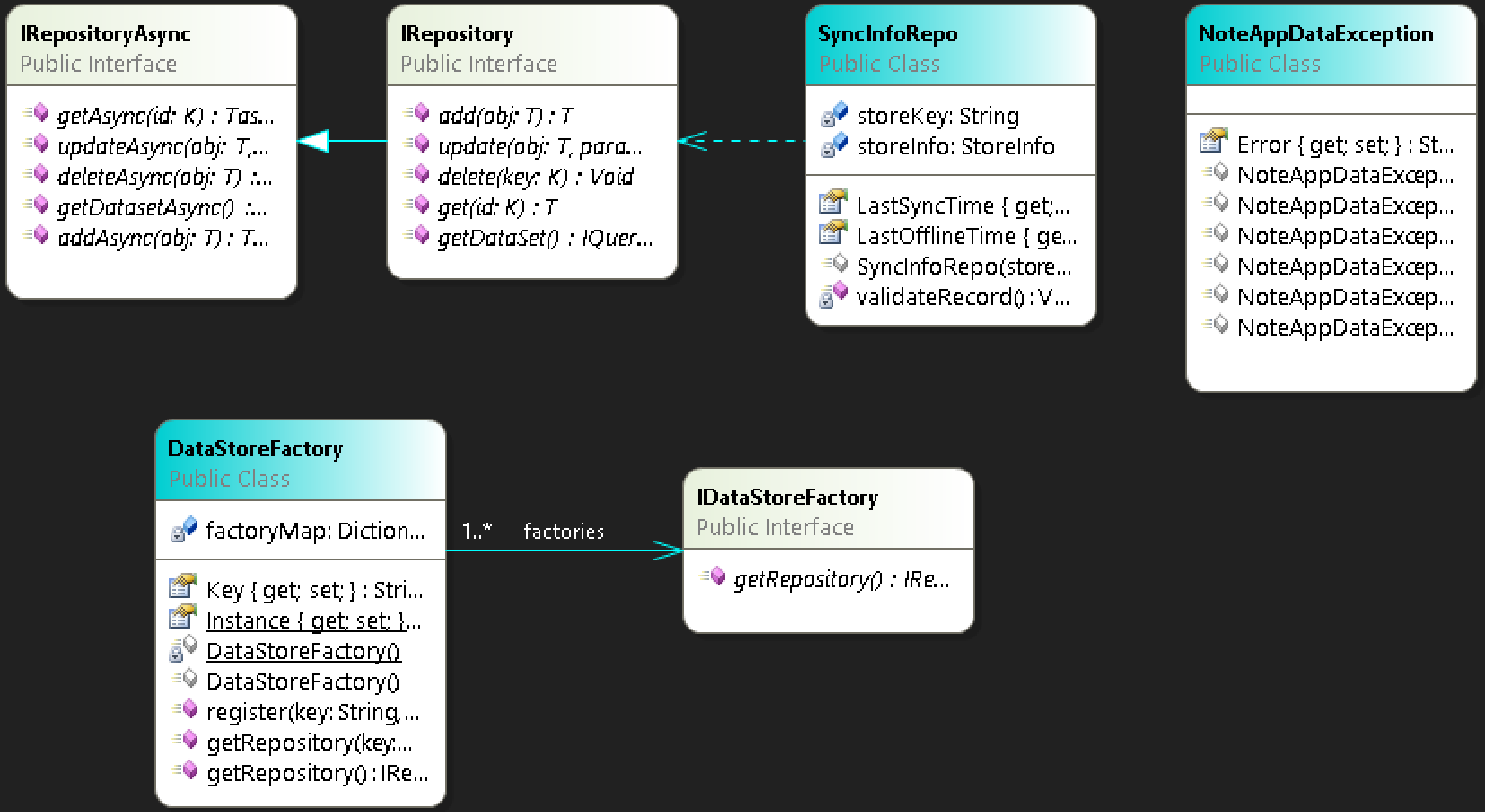
key: String
_user: User

RepositoryKey { get;...
noteService { get; se...
bookService { get; s...
userService { get; se...
user { get; set; } : User
NoteAppService()
NoteAppService(re...
initServices() : Void

IDataService< T, K >
Public Interface

add(obj: T) : T
update(obj: T) : Void
get(key: K) : T
delete(key: K) : Void

NoteAppCore(Persistence)



IRepositoryAsync
Public Interface
- getAsync(id: K) : Tas...
- updateAsync(obj: T,...
- deleteAsync(obj: T) :...
- getDatasetAsync() :...
- addAsync(obj: T) : T...

IRepository
Public Interface
- add(obj: T) : T
- update(obj: T, para...
- delete(key: K) : Void
- get(id: K) : T
- getDataSet() : IQuer...

SyncInfoRepo
Public Class
- storeKey: String
- storeInfo: StoreInfo
- LastSyncTime { get;...
- LastOfflineTime { ge...
- SyncInfoRepo(store...
- validateRecord() : V...

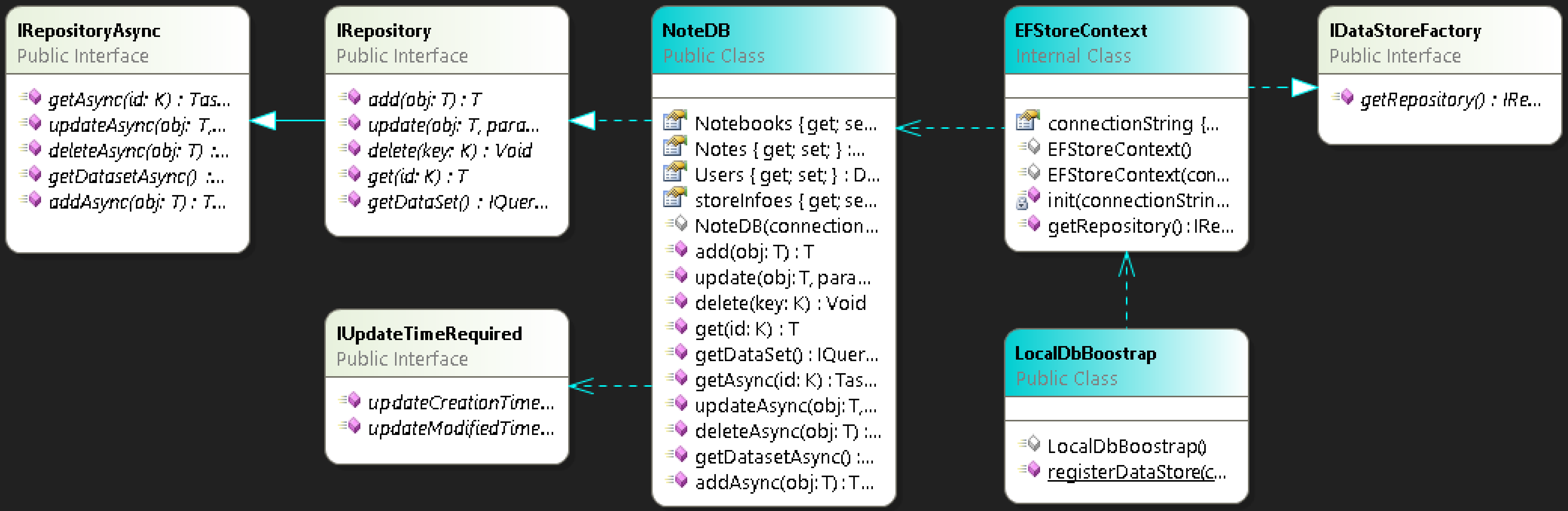
NoteAppDataException
Public Class
- Error { get; set; } : St...
- NoteAppDataExcep...
- NoteAppDataExcep...
- NoteAppDataExcep...
- NoteAppDataExcep...
- NoteAppDataExcep...
- NoteAppDataExcep...

DataStoreFactory
Public Class
- factoryMap: Diction...
- Key { get; set; } : Stri...
- Instance { get; set; }...
- DataStoreFactory()
- DataStoreFactory()
- register(key: String, ...
- getRepository(key:...
- getRepository() : IRe...

1..* factories

IDataStoreFactory
Public Interface
- getRepository() : IRe...

NoteApp(Database)



IRepositoryAsync
Public Interface

- *getAsync(id: K) : Tas...*
- *updateAsync(obj: T,...*
- *deleteAsync(obj: T) :...*
- *getDatasetAsync() :...*
- *addAsync(obj: T) : T...*

IRepository
Public Interface

- *add(obj: T) : T*
- *update(obj: T, para...*
- *delete(key: K) : Void*
- *get(id: K) : T*
- *getDataSet() : IQuer...*

IUpdateTimeRequired
Public Interface

- *updateCreationTime...*
- *updateModifiedTime...*

NoteDB
Public Class

- Notebooks { get; se...
- Notes { get; set; } :...
- Users { get; set; } : D...
- storeInfoes { get; se...
- NoteDB(connection...
- add(obj: T) : T
- update(obj: T, para...
- delete(key: K) : Void
- get(id: K) : T
- getDataSet() : IQuer...
- getAsync(id: K) : Tas...
- updateAsync(obj: T,...
- deleteAsync(obj: T) :...
- getDatasetAsync() :...
- addAsync(obj: T) : T...

EFStoreContext
Internal Class

- connectionString {...
- EFStoreContext()
- EFStoreContext(con...
- init(connectionStrin...
- getRepository() : IRe...

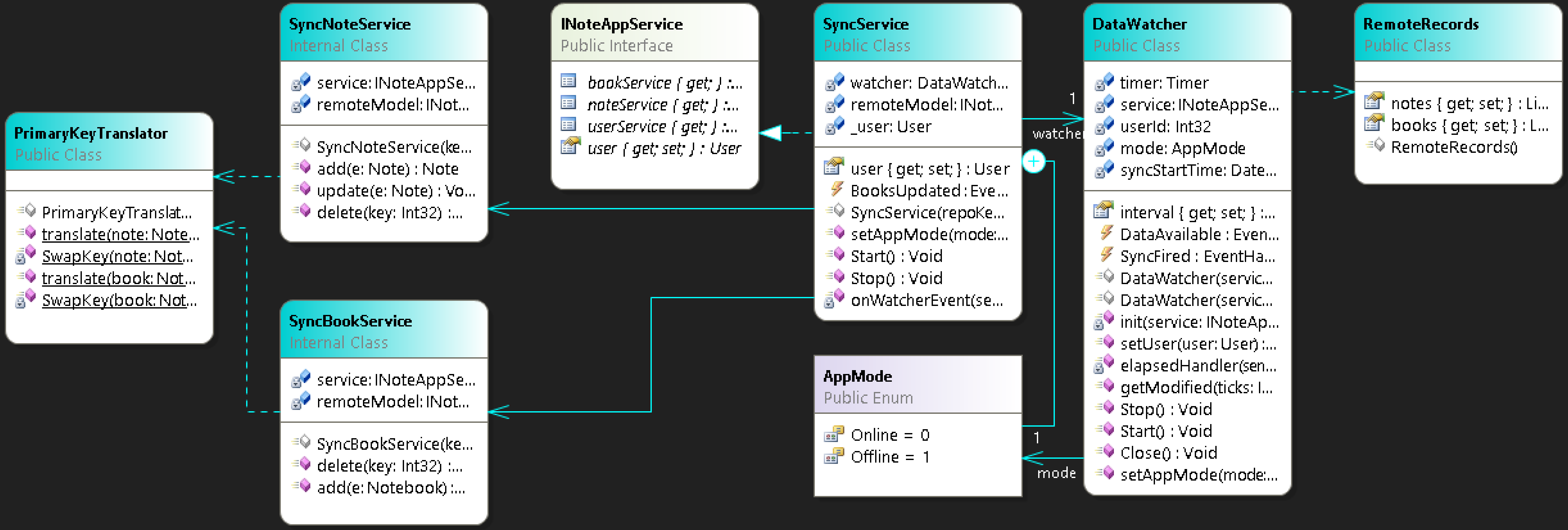
IDataStoreFactory
Public Interface

- *getRepository() : IRe...*

LocalDbBoostrap
Public Class

- LocalDbBoostrap()
- registerDataStore(c...

Sync



PrimaryKeyTranslator
Public Class

- PrimaryKeyTranslat...
- translate(note: Note...
- SwapKey(note: Not...
- translate(book: Not...
- SwapKey(book: Not...

SyncNoteService
Internal Class

- service: INoteAppSe...
- remoteModel: INot...

- SyncNoteService(ke...
- add(e: Note) : Note
- update(e: Note) : Vo...
- delete(key: Int32) :...

SyncBookService
Internal Class

- service: INoteAppSe...
- remoteModel: INot...

- SyncBookService(ke...
- delete(key: Int32) :...
- add(e: Notebook) :...

INoteAppService
Public Interface

- *bookService { get; } :...*
- *noteService { get; } :...*
- *userService { get; } :...*
- *user { get; set; } : User*

SyncService
Public Class

- watcher: DataWatch...
- remoteModel: INot...
- _user: User

- user { get; set; } : User
- BooksUpdated : Eve...
- SyncService(repoKe...
- setAppMode(mode:...
- Start() : Void
- Stop() : Void
- onWatcherEvent(se...

AppMode
Public Enum

- Online = 0
- Offline = 1

DataWatcher
Public Class

- timer: Timer
- service: INoteAppSe...
- userId: Int32
- mode: AppMode
- syncStartTime: Date...

- interval { get; set; } :...
- DataAvailable : Even...
- SyncFired : EventHa...
- DataWatcher(servic...
- DataWatcher(servic...
- init(service: INoteAp...
- setUser(user: User) :...
- elapsedHandler(sen...
- getModified(ticks: I...
- Stop() : Void
- Start() : Void
- Close() : Void
- setAppMode(mode:...

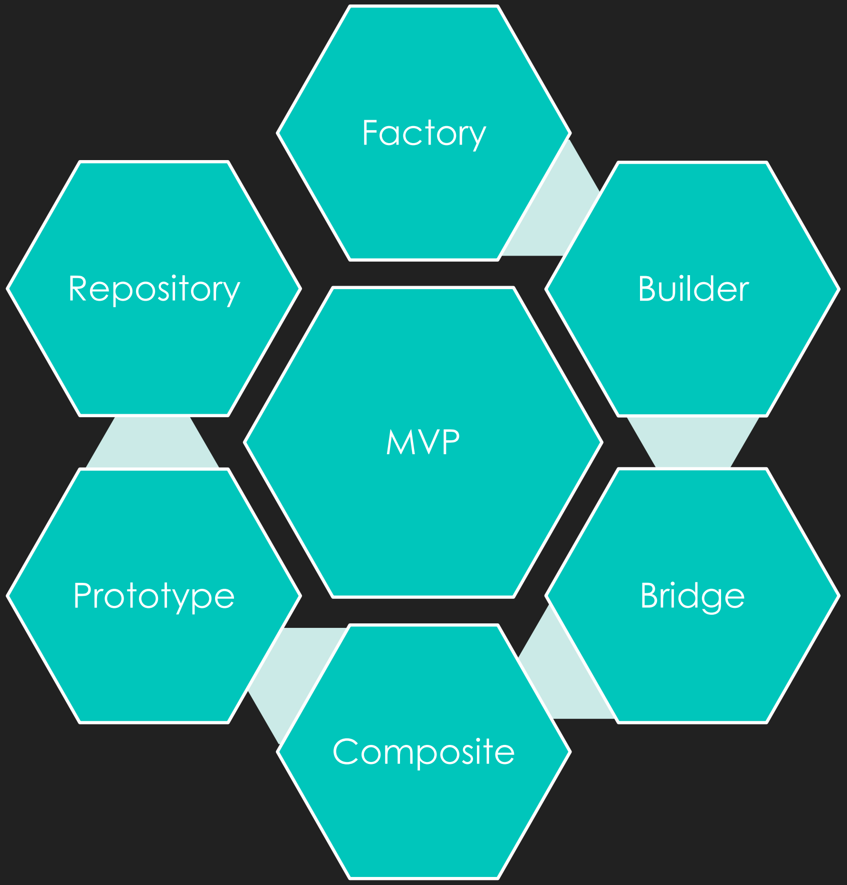
RemoteRecords
Public Class

- notes { get; set; } : Li...
- books { get; set; } : L...
- RemoteRecords()

Bootstrap

- Creates all the dependencies
- Define deployment recipes
- Recipe defined for Windows client
- Call to this bootstrap is the first point of starting the control flow

Design patterns used



MVP

Factory

Builder

Bridge

Composite

Prototype

Repository

Code repository

Repo Link	https://github.com/prajsekar/noteapp
Total number of files	107
Total lines of code	11512
Number of .cs files	73
Lines of code for .cs files	5118

Frameworks used

Component	Framework
UI	WinForms
Local DB	SQL Server Compact
Cloud DB	Azure DocumentDB
ORM	Entity Framework
UT	NUnit
Mock	FakeItEasy

Thank you

Rajasekar Pandiyan