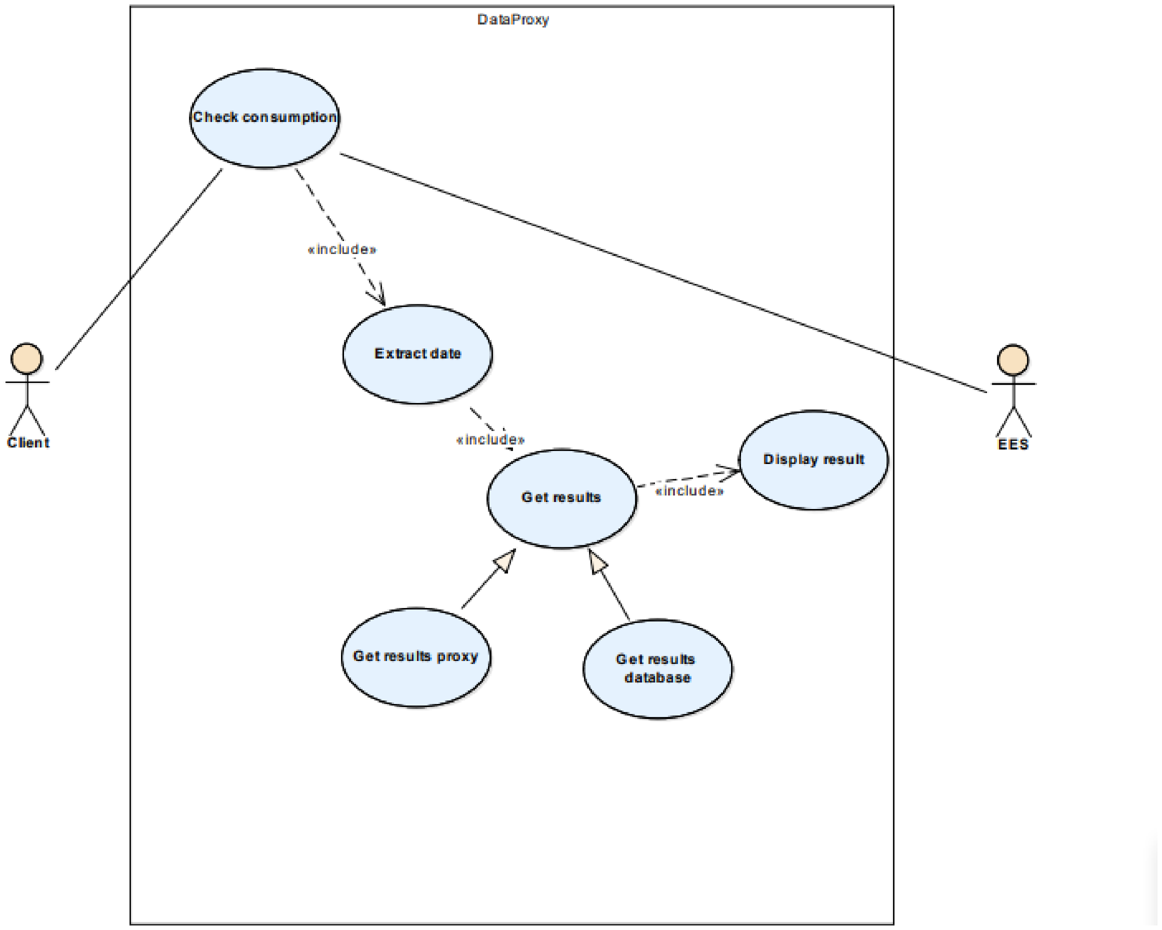
DOCUMENTATION FOR POWER CONSUMPTION DATA

DESING AND ARCHITECTURE BY D. NESTOROV AND N. KARAKLIC

1. Use-Case diagram

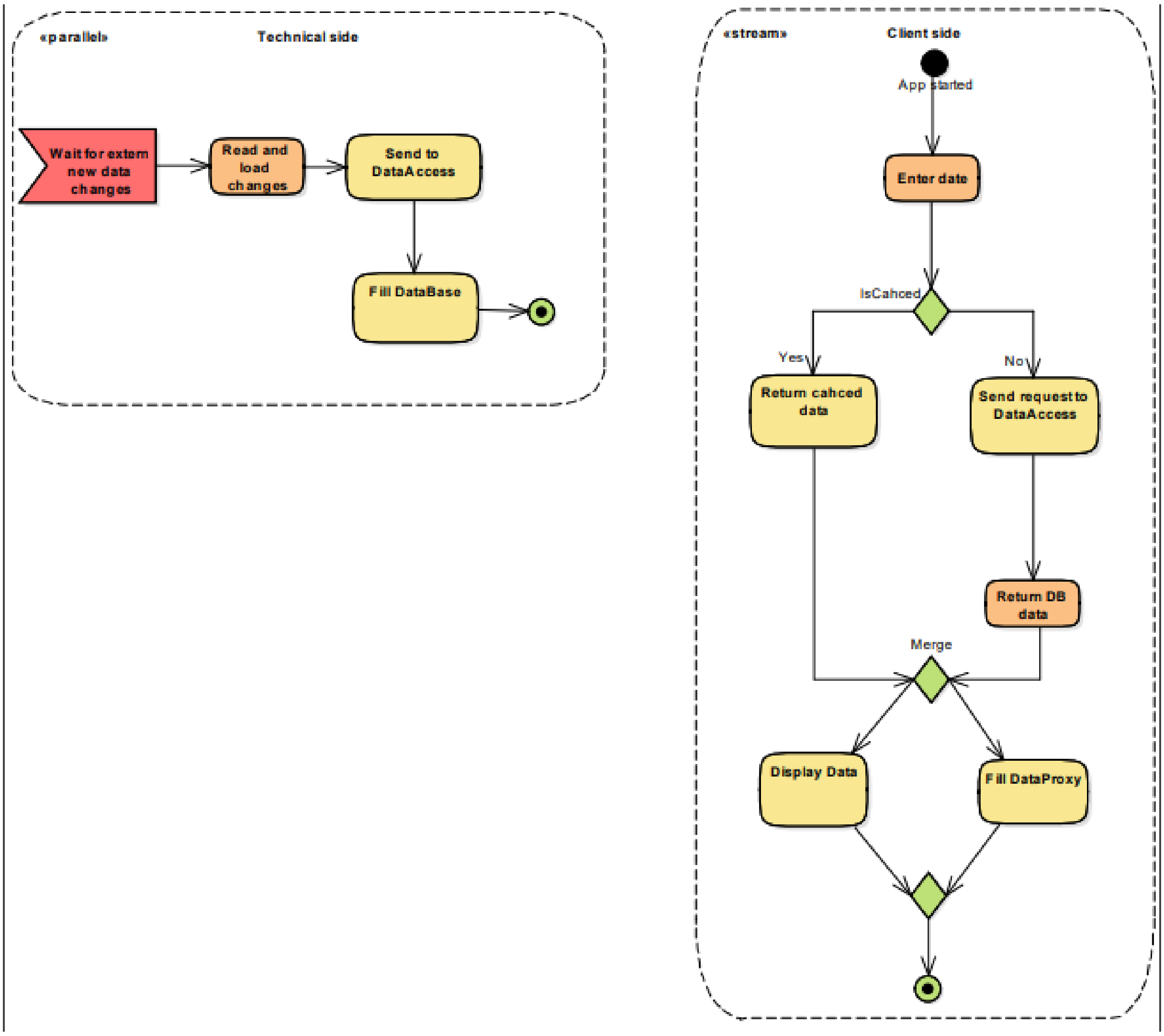
On the diagram below is shown the Use-Case of the power consumption data application. As it is shown, the shared use case is "Check consumption" which is an action performed by both Client and EES system. The idea is that client at any given moment will request, so the server AKA EES system shall deliver it to the client for required date. Further, the system includes sub use cases which are of technical nature.



Next step which is performed by the program is extracting the date from UI. The app will check entered date by client and will get the data needed from Database or "DataProxy" known as cache. Two option are available, one in which the requested data isn't cached, so the app needs to request data directly from DB; and the other case in which the data is stored in the cached memory. Either way, final step is providing the data to the UI and showing it to the Client.

2. Activity diagram

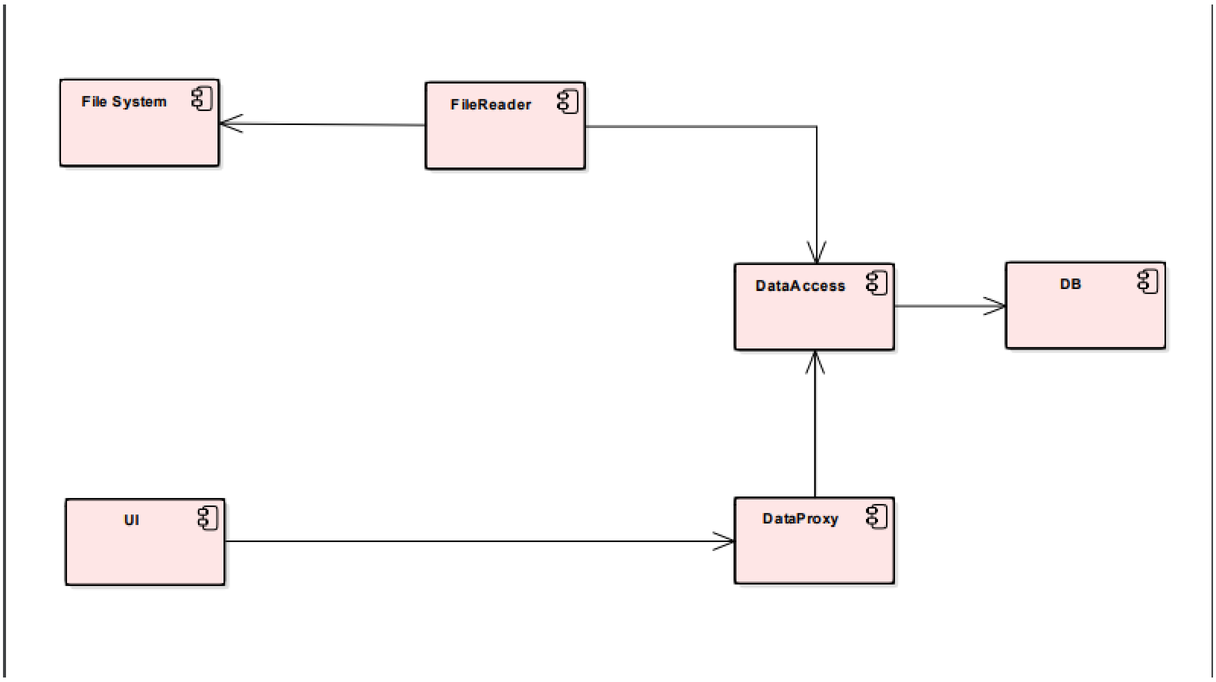
The activity diagram, as it name suggests, is there to provide users with information about steps in the execution process of the app. Two perspectives are shown. One from the clients side, the other is rather technical. As you can see on picture below.



The technical side is an autonomous process which is there to fill the Database as changes in the extern File System are detected. However, the clients side is the part in which the process begins with action “Request date” and is further followed by business logic provided by the developers. All concrete actions and cases are shown the diagram above.

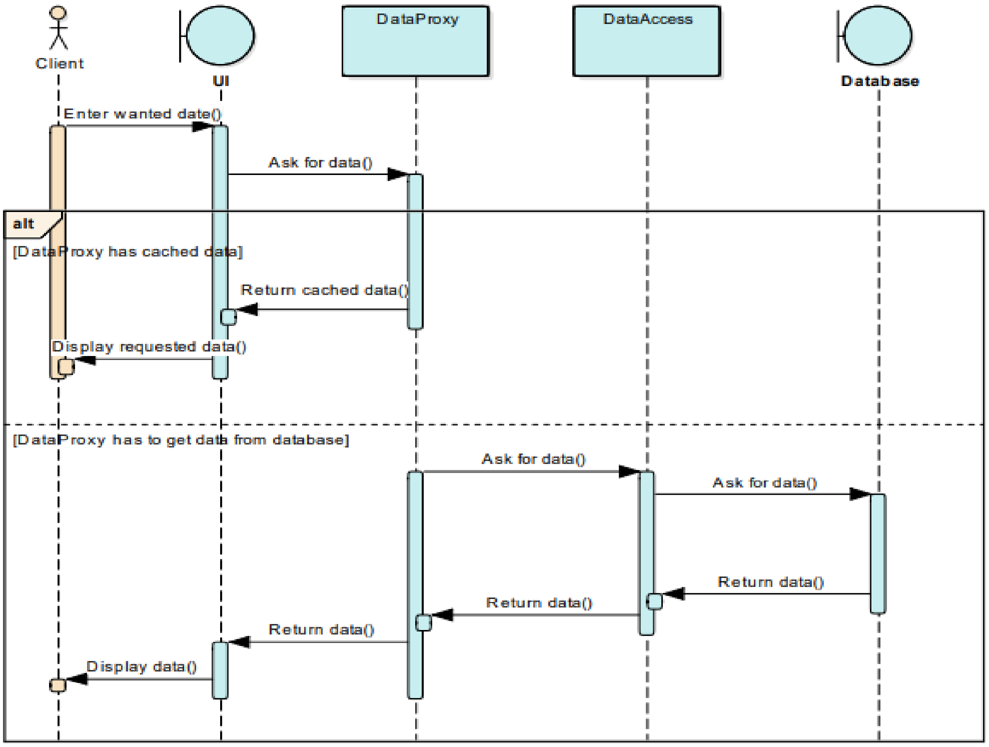
3. Component diagram

The component diagram is based on the assignment given to us. It possesses modules, and every specific module by itself is a component. The information about every component is described in ReadMe, which is to be found on our https://github.com/hodoje/res. The component diagram is shown below.



4. Sequence diagram

The sequence diagram shown below is there to provide users with information in how and which sequence will the action be performed. As from client’s side, in which the client enters the date and provides the app with it. In case it is cached, the data will be provided to the client right away. In other case, which is the technical side it will further request the data from Database, store it in cache, and then provide to the client.



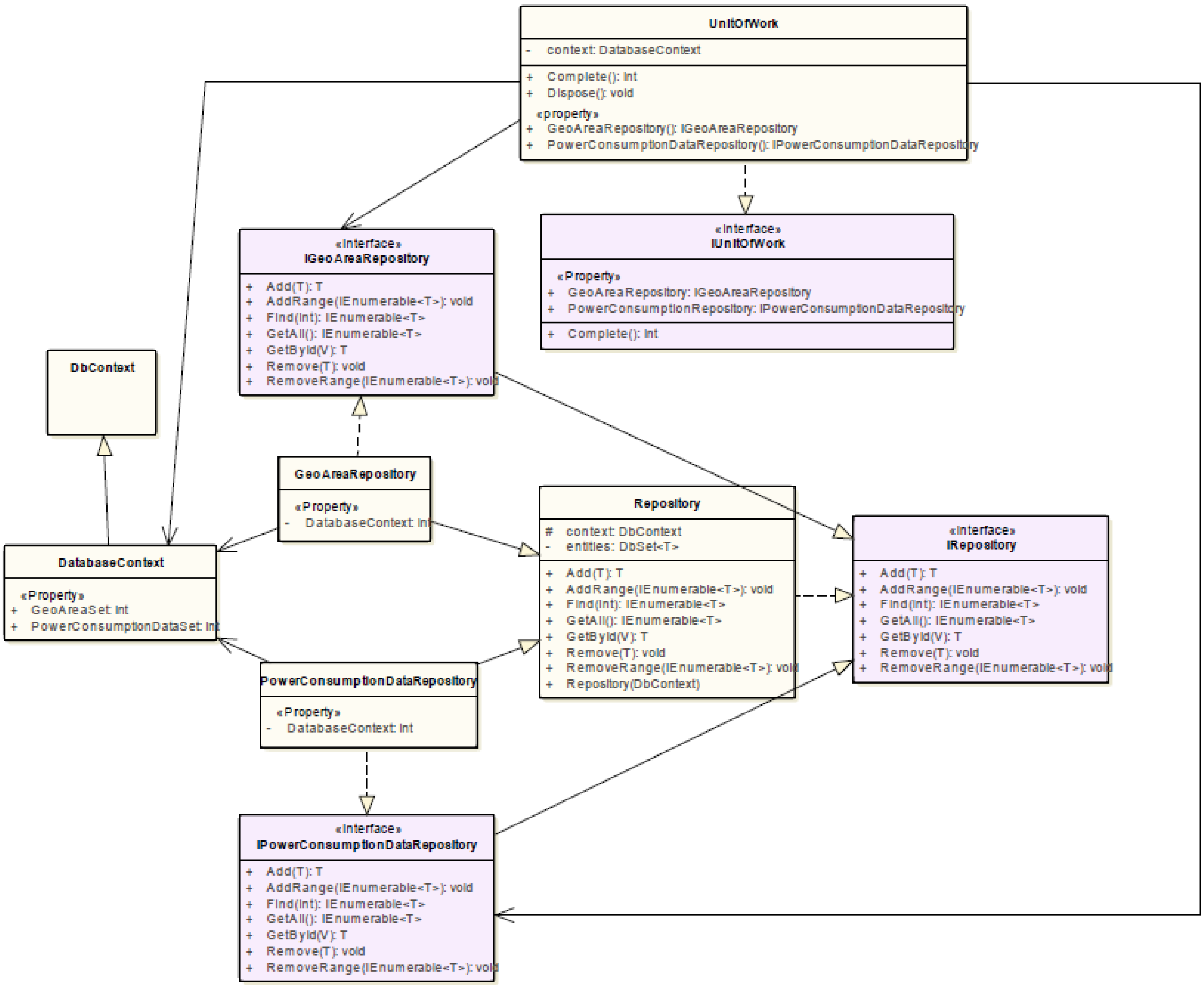
5. Class diagram

Class diagram is consisted of 5 diagrams.

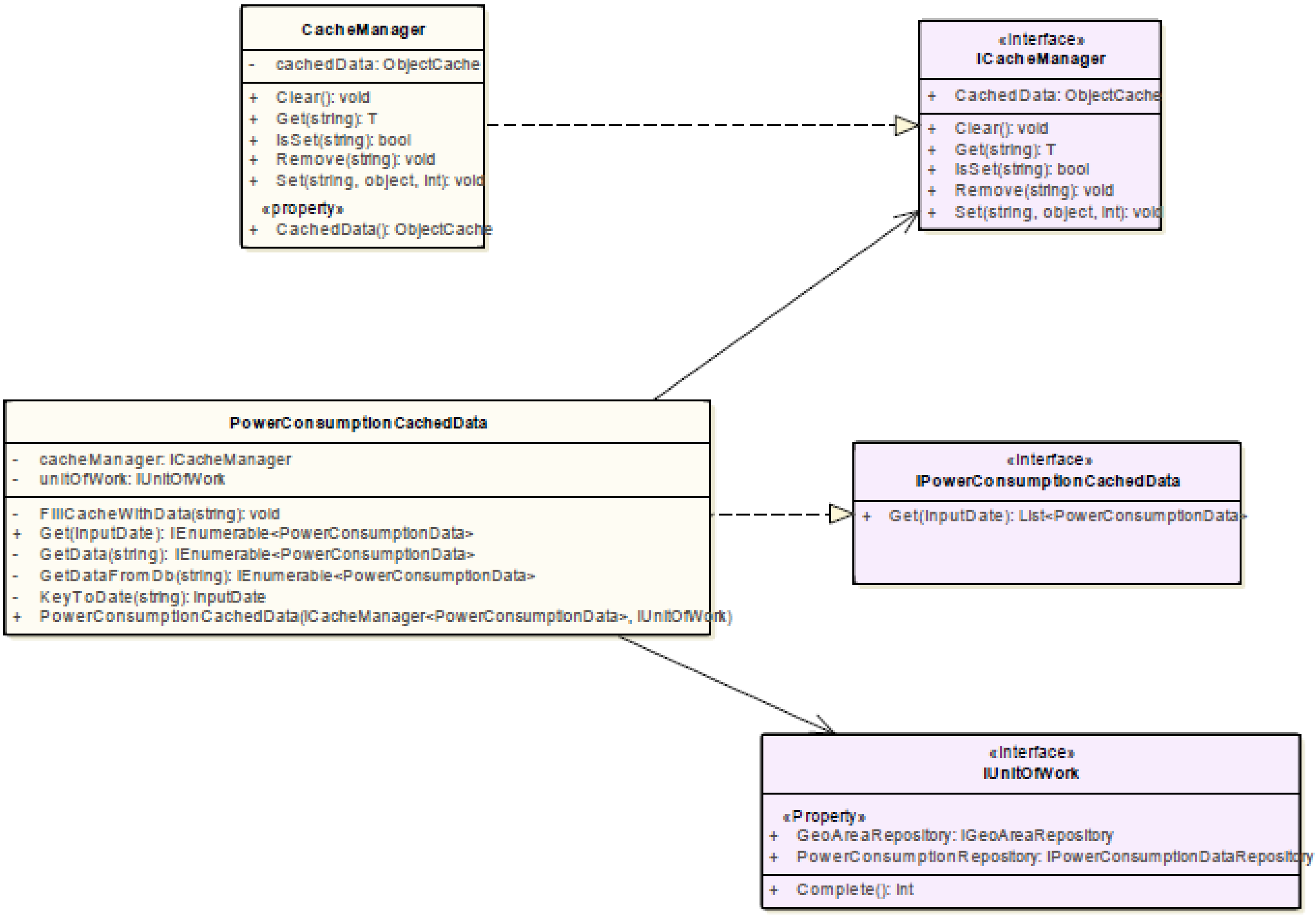
- **Data Access**
- **Data Proxy**
- **File Reader**
- **UI**
- **Entities**

NOTE: Same classes can be used in some diagrams because of the coupling between each VS project.

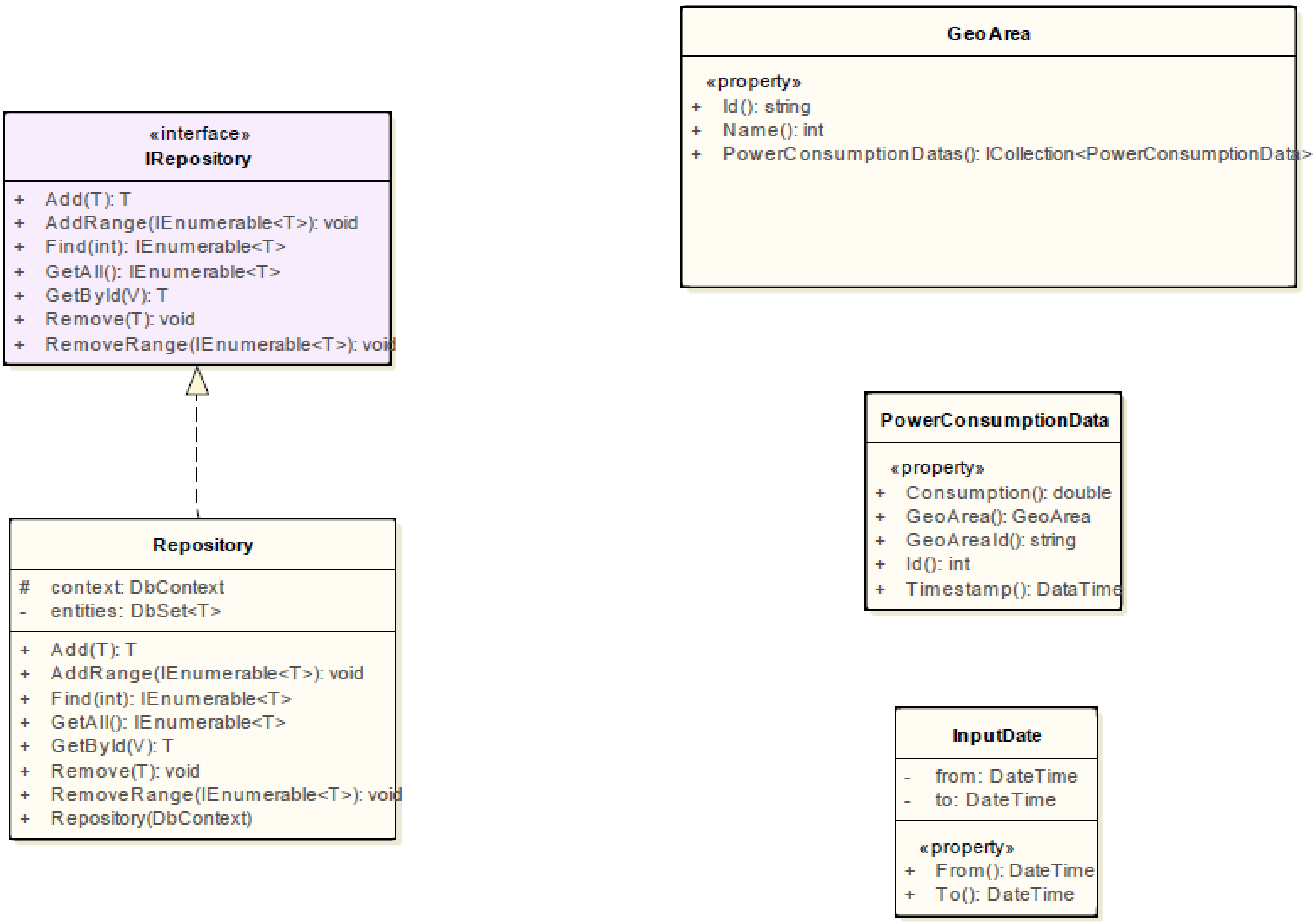
Data Access:


UnitOfWork
- context: DatabaseContext
+ Complete(): int
+ Dispose(): void
«property»
+ GeoAreaRepository(): IGeoAreaRepository
+ PowerConsumptionDataRepository(): IPowerConsumptionDataRepository
«interface»
IUnitOfWork
«Property»
+ GeoAreaRepository: IGeoAreaRepository
+ PowerConsumptionRepository: IPowerConsumptionDataRepository
+ Complete(): int
«interface»
IGeoAreaRepository
+ Add(T): T
+ AddRange(IEnumerable<T>): void
+ Find(int): IEnumerable<T>
+ GetAll(): IEnumerable<T>
+ GetById(V): T
+ Remove(T): void
+ RemoveRange(IEnumerable<T>): void
DbContext
GeoAreaRepository
«Property»
- DatabaseContext: int
Repository
# context: DbContext
- entities: DbSet<T>
+ Add(T): T
+ AddRange(IEnumerable<T>): void
+ Find(int): IEnumerable<T>
+ GetAll(): IEnumerable<T>
+ GetById(V): T
+ Remove(T): void
+ RemoveRange(IEnumerable<T>): void
+ Repository(DbContext)
«interface»
IRepository
+ Add(T): T
+ AddRange(IEnumerable<T>): void
+ Find(int): IEnumerable<T>
+ GetAll(): IEnumerable<T>
+ GetById(V): T
+ Remove(T): void
+ RemoveRange(IEnumerable<T>): void
DatabaseContext
«Property»
+ GeoAreaSet: int
+ PowerConsumptionDataSet: int
PowerConsumptionDataRepository
«Property»
- DatabaseContext: int
«interface»
IPowerConsumptionDataRepository
+ Add(T): T
+ AddRange(IEnumerable<T>): void
+ Find(int): IEnumerable<T>
+ GetAll(): IEnumerable<T>
+ GetById(V): T
+ Remove(T): void
+ RemoveRange(IEnumerable<T>): void

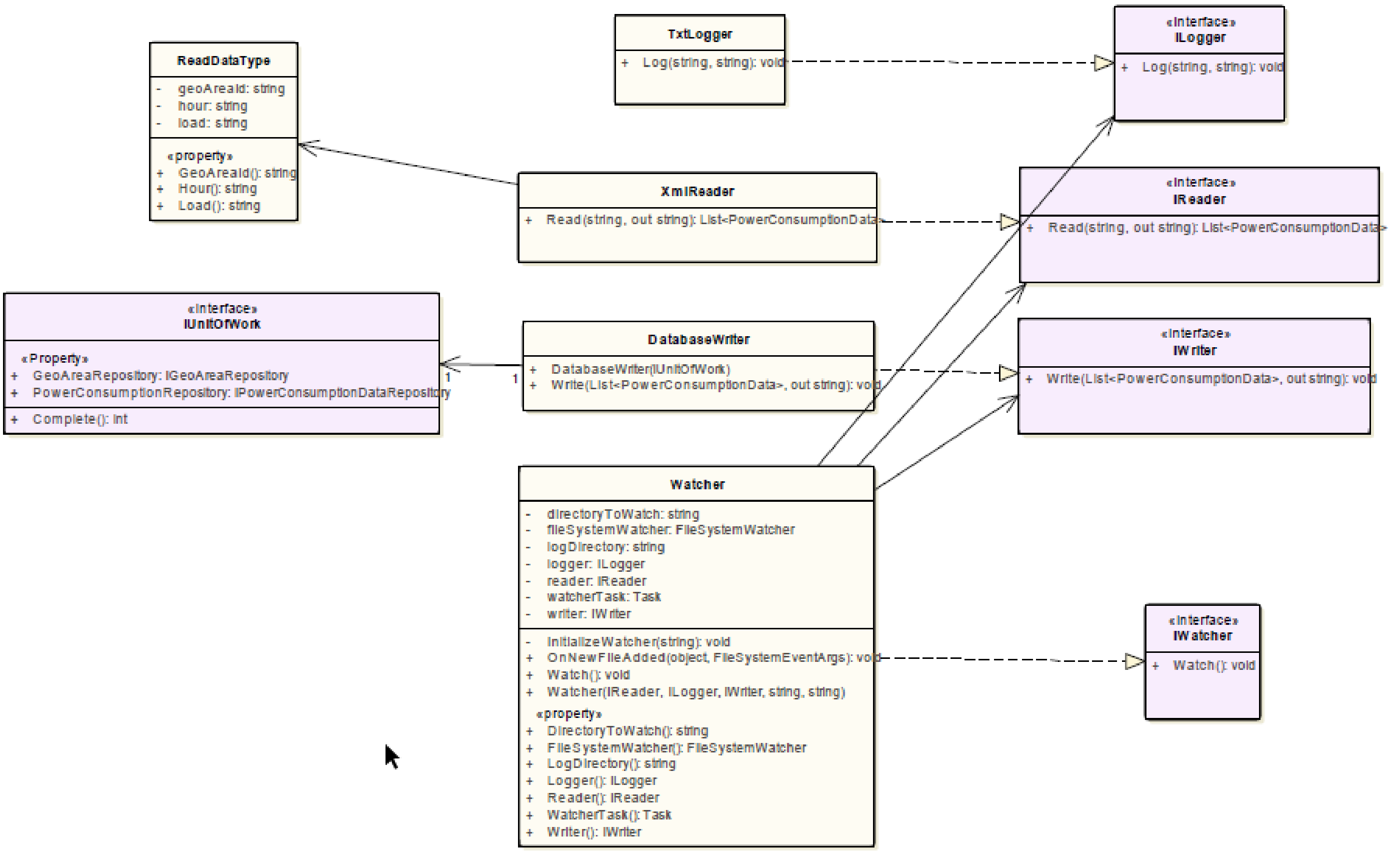
Data Proxy:


CacheManager
- cachedData: ObjectCache
+ Clear(): void
+ Get(string): T
+ IsSet(string): bool
+ Remove(string): void
+ Set(string, object, int): void
«property»
+ CachedData(): ObjectCache
«interface»
ICacheManager
+ CachedData: ObjectCache
+ Clear(): void
+ Get(string): T
+ IsSet(string): bool
+ Remove(string): void
+ Set(string, object, int): void
PowerConsumptionCachedData
- cacheManager: ICacheManager
- unitOfWork: IUnitOfWork
- FillCacheWithData(string): void
+ Get(InputDate): IEnumerable<PowerConsumptionData>
- GetData(string): IEnumerable<PowerConsumptionData>
- GetDataFromDb(string): IEnumerable<PowerConsumptionData>
- KeyToDate(string): InputDate
+ PowerConsumptionCachedData(ICacheManager<PowerConsumptionData>, IUnitOfWork)
«interface»
IPowerConsumptionCachedData
+ Get(InputDate): List<PowerConsumptionData>
«interface»
IUnitOfWork
«Property»
+ GeoAreaRepository: IGeoAreaRepository
+ PowerConsumptionRepository: IPowerConsumptionDataRepository
+ Complete(): int

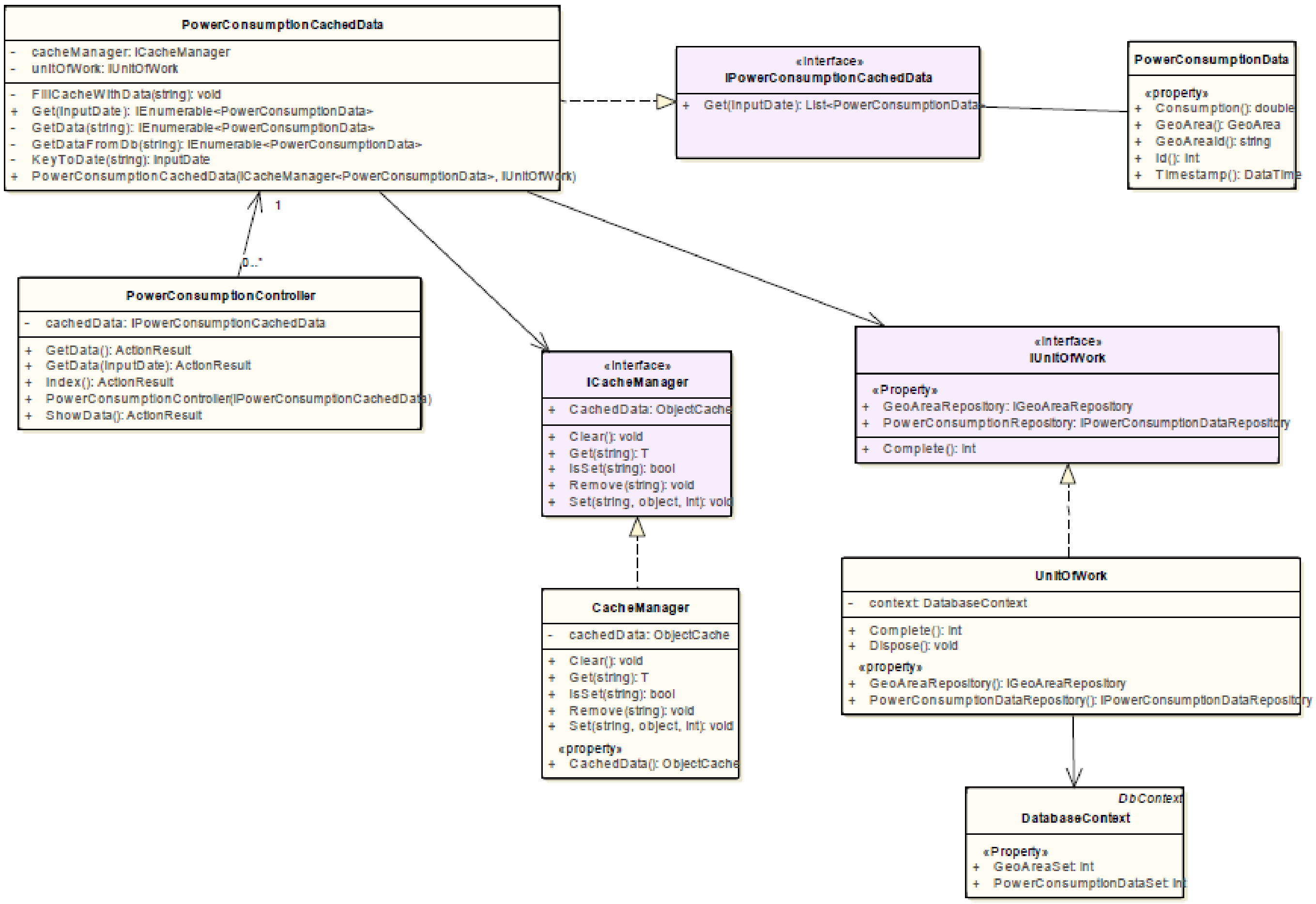
Entities:


«interface»
IRepository
+ Add(T): T
+ AddRange(IEnumerable<T>): void
+ Find(int): IEnumerable<T>
+ GetAll(): IEnumerable<T>
+ GetById(V): T
+ Remove(T): void
+ RemoveRange(IEnumerable<T>): void
Repository
# context: DbContext
- entities: DbSet<T>
+ Add(T): T
+ AddRange(IEnumerable<T>): void
+ Find(int): IEnumerable<T>
+ GetAll(): IEnumerable<T>
+ GetById(V): T
+ Remove(T): void
+ RemoveRange(IEnumerable<T>): void
+ Repository(DbContext)
GeoArea
«property»
+ Id(): string
+ Name(): int
+ PowerConsumptionDatas(): ICollection<PowerConsumptionData>
PowerConsumptionData
«property»
+ Consumption(): double
+ GeoArea(): GeoArea
+ GeoAreaId(): string
+ Id(): int
+ Timestamp(): DataTime
InputDate
- from: DateTime
- to: DateTime
«property»
+ From(): DateTime
+ To(): DateTime

File Reader:


ReadDataType
- geoAreaId: string
- hour: string
- load: string
«property»
+ GeoAreaId(): string
+ Hour(): string
+ Load(): string
TxtLogger
+ Log(string, string): void
«interface»
ILogger
+ Log(string, string): void
XmlReader
+ Read(string, out string): List<PowerConsumptionData>
«interface»
IReader
+ Read(string, out string): List<PowerConsumptionData>
«interface»
IUnitOfWork
«Property»
+ GeoAreaRepository: IGeoAreaRepository
+ PowerConsumptionRepository: IPowerConsumptionDataRepository
+ Complete(): int
1
1
DatabaseWriter
+ DatabaseWriter(IUnitOfWork)
+ Write(List<PowerConsumptionData>, out string): void
«interface»
IWriter
+ Write(List<PowerConsumptionData>, out string): void
Watcher
- directoryToWatch: string
- fileSystemWatcher: FileSystemWatcher
- logDirectory: string
- logger: ILogger
- reader: IReader
- watcherTask: Task
- writer: IWriter
- InitializeWatcher(string): void
+ OnNewFileAdded(object, FileSystemEventArgs): void
+ Watch(): void
+ Watcher(IReader, ILogger, IWriter, string, string)
«property»
+ DirectoryToWatch(): string
+ FileSystemWatcher(): FileSystemWatcher
+ LogDirectory(): string
+ Logger(): ILogger
+ Reader(): IReader
+ WatcherTask(): Task
+ Writer(): IWriter
«interface»
IWatcher
+ Watch(): void

UI:


PowerConsumptionCachedData
- cacheManager: ICacheManager
- unitOfWork: IUnitOfWork
- FillCacheWithData(string): void
+ Get(InputDate): IEnumerable<PowerConsumptionData>
- GetData(string): IEnumerable<PowerConsumptionData>
- GetDataFromDb(string): IEnumerable<PowerConsumptionData>
- KeyToDate(string): InputDate
+ PowerConsumptionCachedData(ICacheManager<PowerConsumptionData>, IUnitOfWork)
«interface»
IPowerConsumptionCachedData
+ Get(InputDate): List<PowerConsumptionData>
PowerConsumptionData
«property»
+ Consumption(): double
+ GeoArea(): GeoArea
+ GeoAreaId(): string
+ Id(): int
+ Timestamp(): DataTime
1
0..*
PowerConsumptionController
- cachedData: IPowerConsumptionCachedData
+ GetData(): ActionResult
+ GetData(InputDate): ActionResult
+ Index(): ActionResult
+ PowerConsumptionController(IPowerConsumptionCachedData)
+ ShowData(): ActionResult
«interface»
ICacheManager
+ CachedData: ObjectCache
+ Clear(): void
+ Get(string): T
+ IsSet(string): bool
+ Remove(string): void
+ Set(string, object, int): void
CacheManager
- cachedData: ObjectCache
+ Clear(): void
+ Get(string): T
+ IsSet(string): bool
+ Remove(string): void
+ Set(string, object, int): void
«property»
+ CachedData(): ObjectCache
«interface»
IUnitOfWork
«Property»
+ GeoAreaRepository: IGeoAreaRepository
+ PowerConsumptionRepository: IPowerConsumptionDataRepository
+ Complete(): int
UnitOfWork
- context: DatabaseContext
+ Complete(): int
+ Dispose(): void
«property»
+ GeoAreaRepository(): IGeoAreaRepository
+ PowerConsumptionDataRepository(): IPowerConsumptionDataRepository
DbContext
DatabaseContext
«Property»
+ GeoAreaSet: int
+ PowerConsumptionDataSet: int