1. Technical description

# . General description

DataStorage solution consists of four projects:

* **DataLibrary** – classes and interfaces that help to interact with database to get and modify data
* **DataStorageLibrary** – classes and interfaces that defines data objects and possible actions to do on that objects
* **DataStorageWebAPI** – classes that defines web services that let to execute data storage actions
* **DataStorageCLI** – defines console application that let to execute data storage actions

# . Projects details

Solution consists of the following elements:

1. **Connection provider**

Provides connection to database. Is represented by the following interface:

IConnectionProvider

IDbConnection GetConnection();

Classes that implements interface:

BaseConnectionProvider – abstract class for storing connection strings

PgConnectionProvider – provides connection to PostgreSql database

1. **Action**

Defines action logic. Is represented by the following interface:

IAction<T>

T Execute(params IParam[] parameters);

Classes that implements interface:

DbCommandAction – abstract class for actions that uses IDbConnection and IDbCommand interfaces to interact with database

GetSingleCommandAction – class for DbCommand actions that returns single result

GetMultipleCommandAction – class for DbCommand actions that returns multiple results

NonQueryCommandAction – class for DbCommand actions that modifies data

1. **Param**

Defines action parameter. Is represented by the following interface:

IParam

String Name { get; }

object Value { get; }

1. **Param mapper**

Maps input objects to action parameters. Is represented by the following interface:

IParamMapper<T>

IParam[] MapToParams(T param);

Classes that implements interface:

AddBookParamMapper – maps book to parameters for addition of book to database

1. **Result mapper**

Maps returned database record data to objects. Is represented by the following interface:

IResultMapper<T>

T MapResult(IDataRecord dataRecord);

Classes that implements interface:

IntResultMapper – gets single int value

BookMetadataRM – maps resulted metadata of books record to book object

ContentIndexRM – maps resulted data of content index

1. **Formatter**

Formats data. Is represented by the following interface:

IFormatter<T>

String Format(T input);

Classes that implements interface:

BookMetadataFormatter – formats book metadata

ContentIndexFormatter – formats content index

1. **Data**

Defines stored data.

Book – Defines book data

ContentIndex – Defines content index data

1. **Storage**

Defines list of actions that can be done on specific data. Is represented by the following interfaces:

ISimpleStorage<T> - represents basic actions

IEnumerable<int> GetAllIds();

T GetById(int id);

IStorage<T>:ISimpleStorage<T> - represents extended common actions

void RemoveById(int id);

void Add(T item);

IBookStorage:ISimpleStorage<T> – represents actions possible for books

List<String> GetTableOfContent(int id);

Classes that implements interface:

SimpleStorage – implementation of IsimpleStorage that uses Iaction interfaces for action execution

**Storage –** implementation of IStorage that uses IAction interfaces for action execution

**BookStorage -** implementation of IBookStorage that uses Iaction interfaces for action execution

1. **Controller**

Defines web services that let to execute data storage actions. Solution contains the following controllers:

SimpleStorageController – defines web services for basic actions

StorageController – defines web services for extended common actions

BooksController – defines web services for actions possible for books