SDV701 ASSINGMENT 2

Stage 2

Oleg Sivers

Project git: https://github.com/sio2k1/A2

Contents

Project on GitHub

Link to repository: https://github.com/sio2k1/A2

Layers, Tiers and Architectural Patterns

Layers and Tiers

We are building 3-layer system including presentation, business and data. We have two applications for presentation layer – Windows and WEB clients, business layer is spread between our back-end code and database stored procedures, data layer is MS SQL server database. Layers is a way to logically organize the code, while tiers are about where the code is physically executed. Currently my design based on two tiers:

- Client applications
- Server gRPC API and database

In the future database part can be put into separate server, forming another tier and reducing overall load on application server, but increasing traffic flow.

As we can separate business and data layers from presentation, we can setup multiple clients and we also can add another client, for mobile platforms, for example. Layered and tiered applications are harder to develop and debug, but flexibility of such structures is great. Having an API is also affecting data security, as user application does not get an access to database directly, it can perform only certain API calls, which are translated to database queries at the back-end. We can say that its better security-wise. We also can get an access to multiple databases, using one API, if its required.

Architectural Patterns

Data transfer object – we are using several classes to transfer data within its objects between server and client, these classes are automatically generated by gRPC, definition can be found in netsop.proto file. Example:

```
1.  message GetJsonSerializedResponse {
2.    string response = 1;
3.    string errorMessage = 2;
4.  }
```

In our design GRPC DTO objects are carrying serialized objects of model classes, which is one of possible scenarios.

Data Mapper – I'm not completely follow this particular design pattern, and breaking the rules here, my data mapping solution relays on similarities between field names in tables and public properties in classes. Mapping implemented as generic method to suite almost all tables that store object data. Result of mapping is a list of objects of type, provided as a generic parameter. Mapper code for selection / non query can be found in "SDV701BackEnd.DB.sqlDBExecuter". This solution is really flexible, as we don't have to write any particular mapping code for concrete classes, but we still need to consider database changes and field renaming deleting, as it can break mapping.

Insert, update and delete queries based on object field values are generated by "SDV701BackEnd.infrastructure.QueryGenerator". There are couple limitations here, classes and table should have Identity field, named "id". Originally, I was using first property in property list as "id" field, but it does not work for sub classes, as property order is different, so solution was to

assume all tables and objects for may mapping system are having “id” field / property. We also need to consider field renaming / deleting can break mapping if not done properly. My data mapping solution is simple to use and flexible, but it is based on .net Reflection and it is not really fast performance-wise, because for each row in record set, we have to compare field names and object properties and set properties values. Database mapper code is physically located inside server-side part of the project, which can cause performance issues under high loads. Still, mapping code is flexible in terms of adding extra features, tables and classes to our solution, as we don’t have to write any extra mapping code.

Single table inheritance – My mapping solution support TPH mapping for subclasses, we have to call different method “MapHierarchy” for mapping TPH, we also have to specify an extra field in table for storing an actual class name.

Service layer – SDV701BackEnd.NetshopService is a service layer in our project, providing gRPC endpoint for our client applications.

State management

Win client

Order list – we loading all available orders, it’s not really good solution, in the future we need to add date and order state filters to load only required / active orders.

Category list – we are loading all available categories, this is perfectly fine, as we have limited set of categories and limited number of queries, as this is admin application.

Category editing – we are loading all items available for current category.

Item editing – we are not loading any data, we are operating with list of objects, obtained from previous query, and only send delete / update / insert requests to API upon actual changes.

Web client

Category list – I’ve created an example of caching solution for my client application. Categories list is a good example to cache, as it won’t be changed often. Server is generating hash of string, which containing all categories names. Client stores categories list and hash in local storage. Each time we accessing categories page client will request hash information, if hash is different or we don’t have any data in local storage client will perform full gRPC request to get categories list, otherwise list will be loaded from local storage.

Item list – we are loading all items available for current category.

Order – depending on app state we either load item info from app state or we request one item info from API, it depends on URL address, if user launch app and follow category -> item -> order, item info would be in app state, but if user directly navigate to URL, saved in history “/order/{itemID}”, application won’t know in this case any info about an item except of an ID. In this case application will do extra request to get particular item’s details.

Concurrency control

Win client

We can delete, deleted items and orders, as nothing wrong with it.

We cannot update items, which displayed in application, but were deleted by another user. We will get a message about it.

Two users can modify the same item, but changes will be saved from last user, who will press “Save” button, this is not really good practice, we have to add some functionality to display that current state in database was changed since editing have started.

Web client

We are using optimistic concurrency control for adding orders and subtract quantity from warehouse:

```
update Npart set QtyInStock=QtyInStock PartsQty
where id=@PartsId and QtyInStock>=@qtyleft
if (@@rowcount = 1) begin
    --insert order
end else
begin
    --raise error
end
```

Before performing conditional update, we are doing simple quantity check, as update operation is more costly performance-wise.

We also limit quantity to order in client application by setting “InputNumber max=[CurrentAvailableQtyInStock]”, so user can’t input more than we have in stock in order form.

gRPC instead of REST

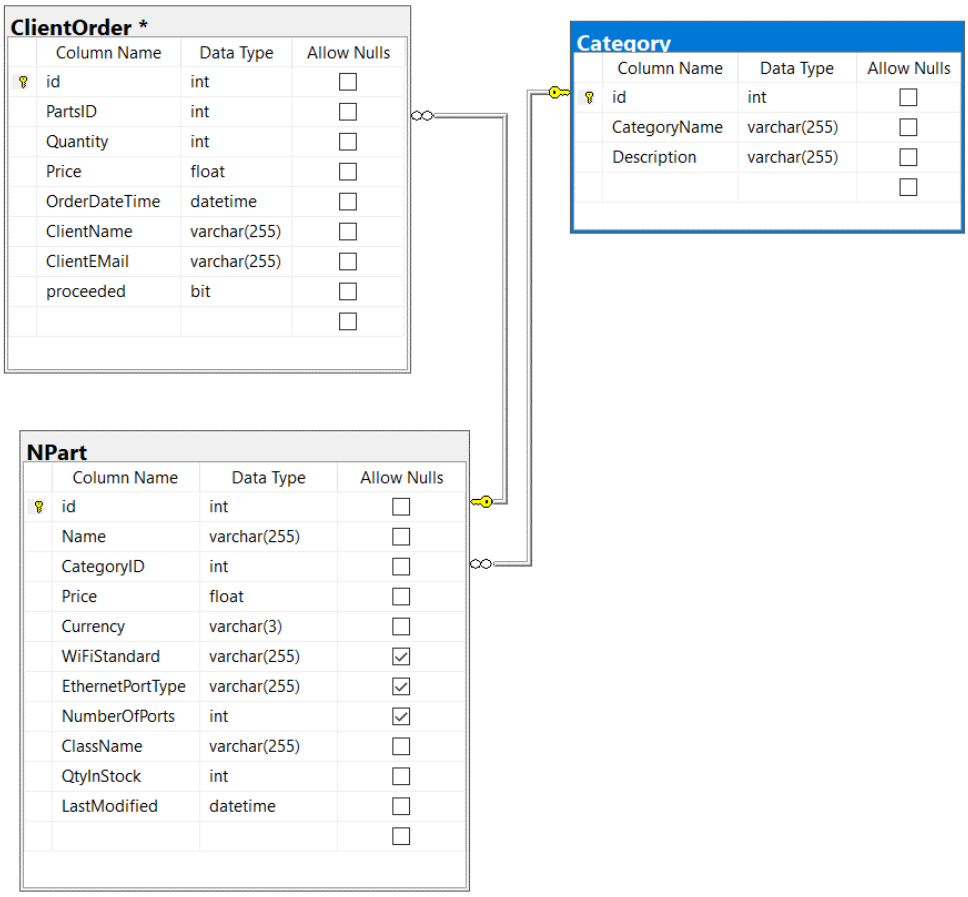
I’m using gRPC for data transfer, overall, I was inspired by NDC conference tutorial, how simple is to implement gRPC in .Net application. Actual solution was not that simple as in tutorial, and I have several issues with my web application, as my web framework is not supporting HTTP/2. But I’ve managed to create shared gRPC client library and use it in both WPF and WEB applications. There are several benefits around using gRPC:

- It has good documentation
- It is supported on different frameworks \ platforms (Including Node.js, .NET)
- It is based on binary blob transfer over HTTP/2 which means it faster and consumes less traffic than REST API
- Server and client define same data transfer format upon compilation, this makes data transfer more solid.
- gRPC massages can have a complex structure, including nesting and arrays.

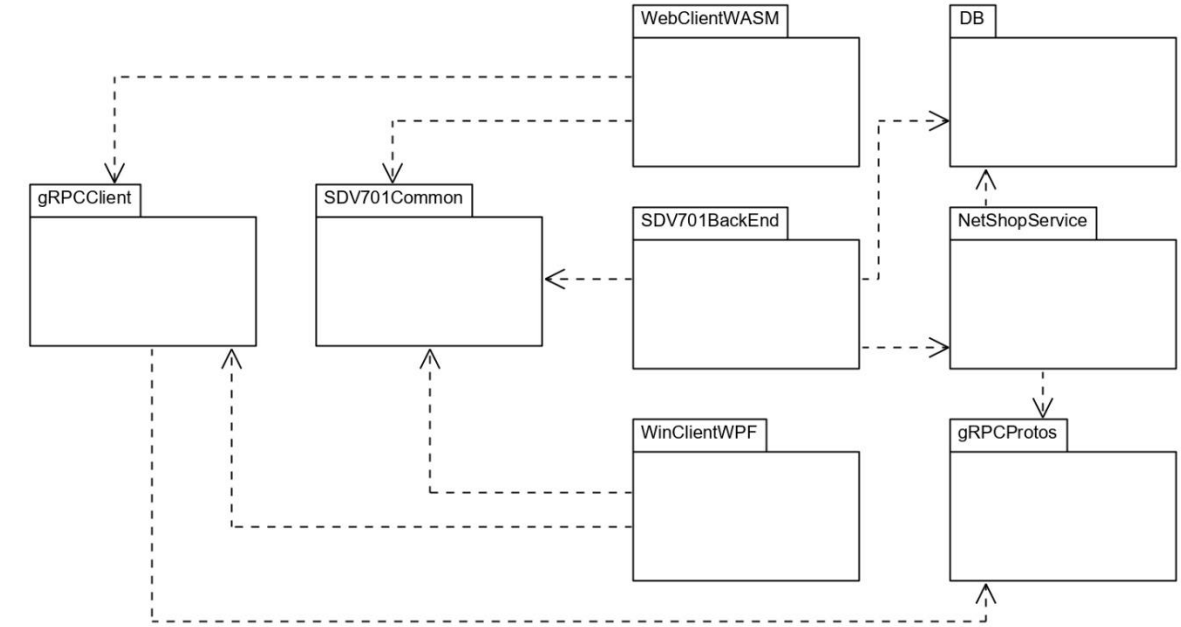
Code of gRPC client code can be found in gRPCClient assembly and server code located in NetshopService.

Diagrams

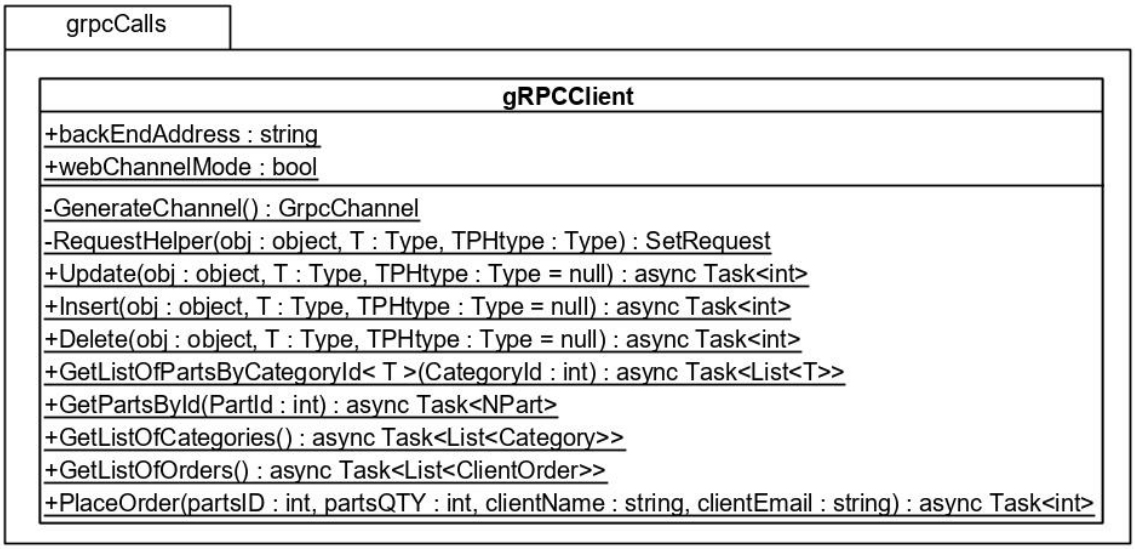
Database



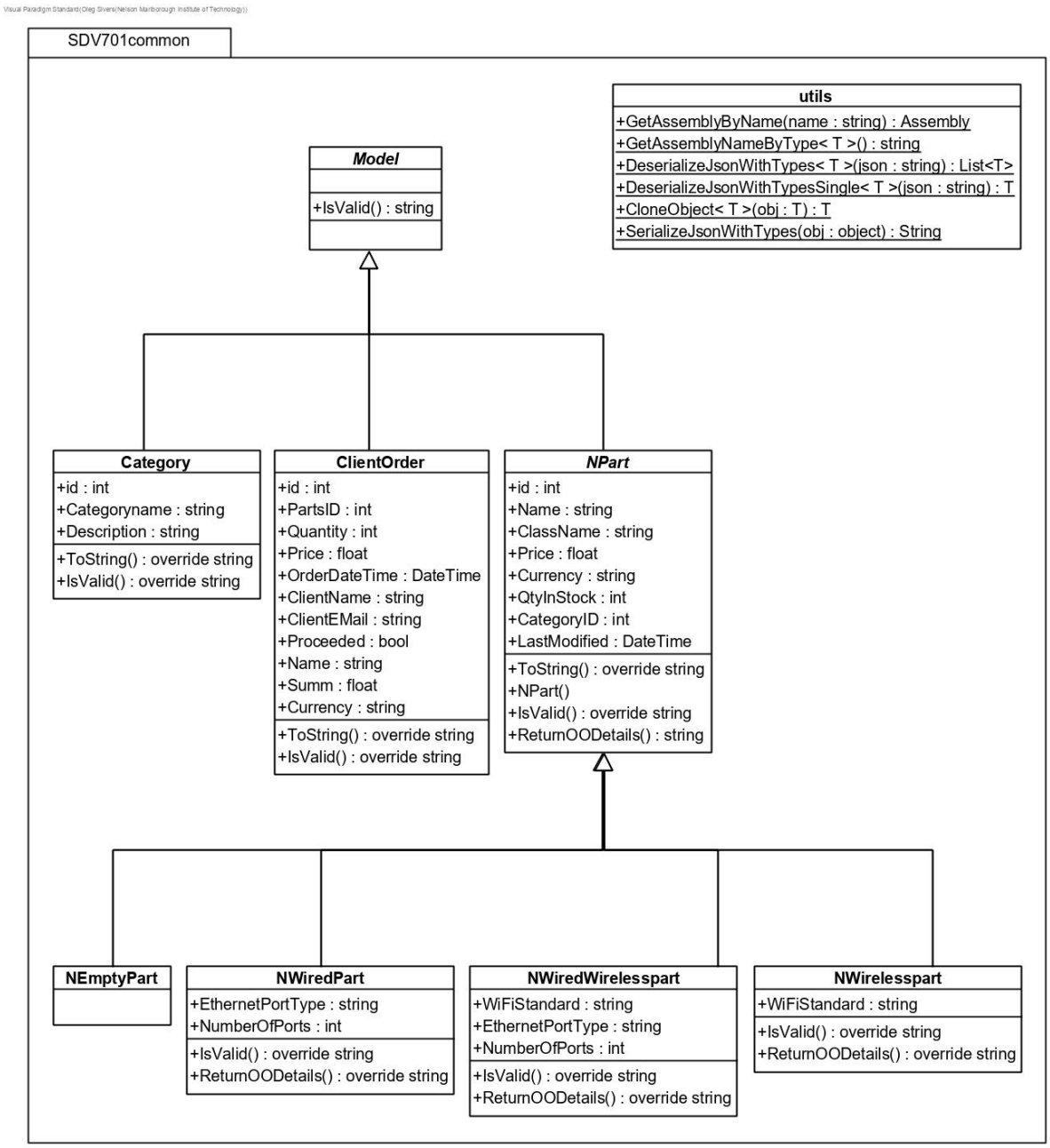
Packages



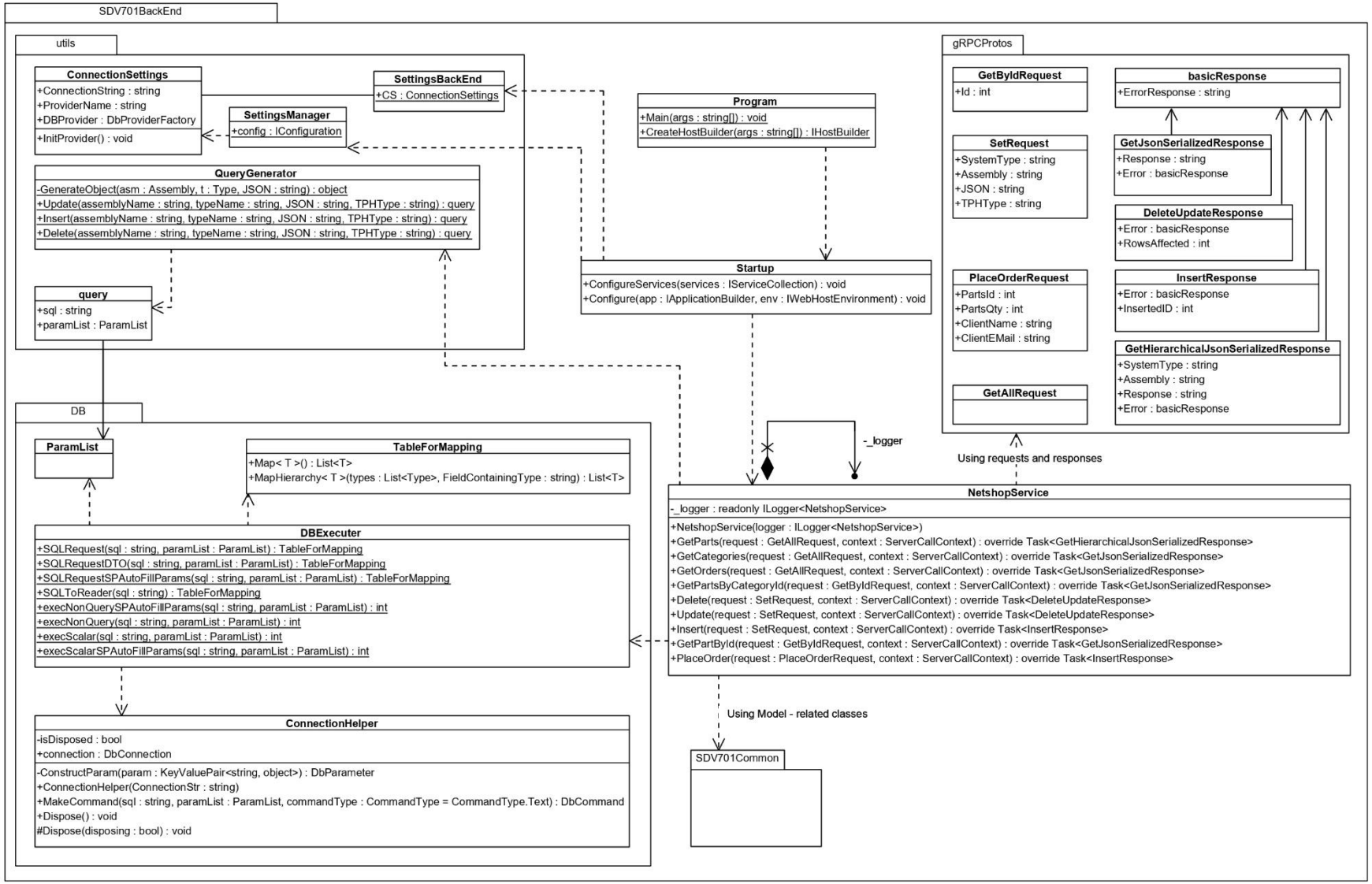
gRPCClient



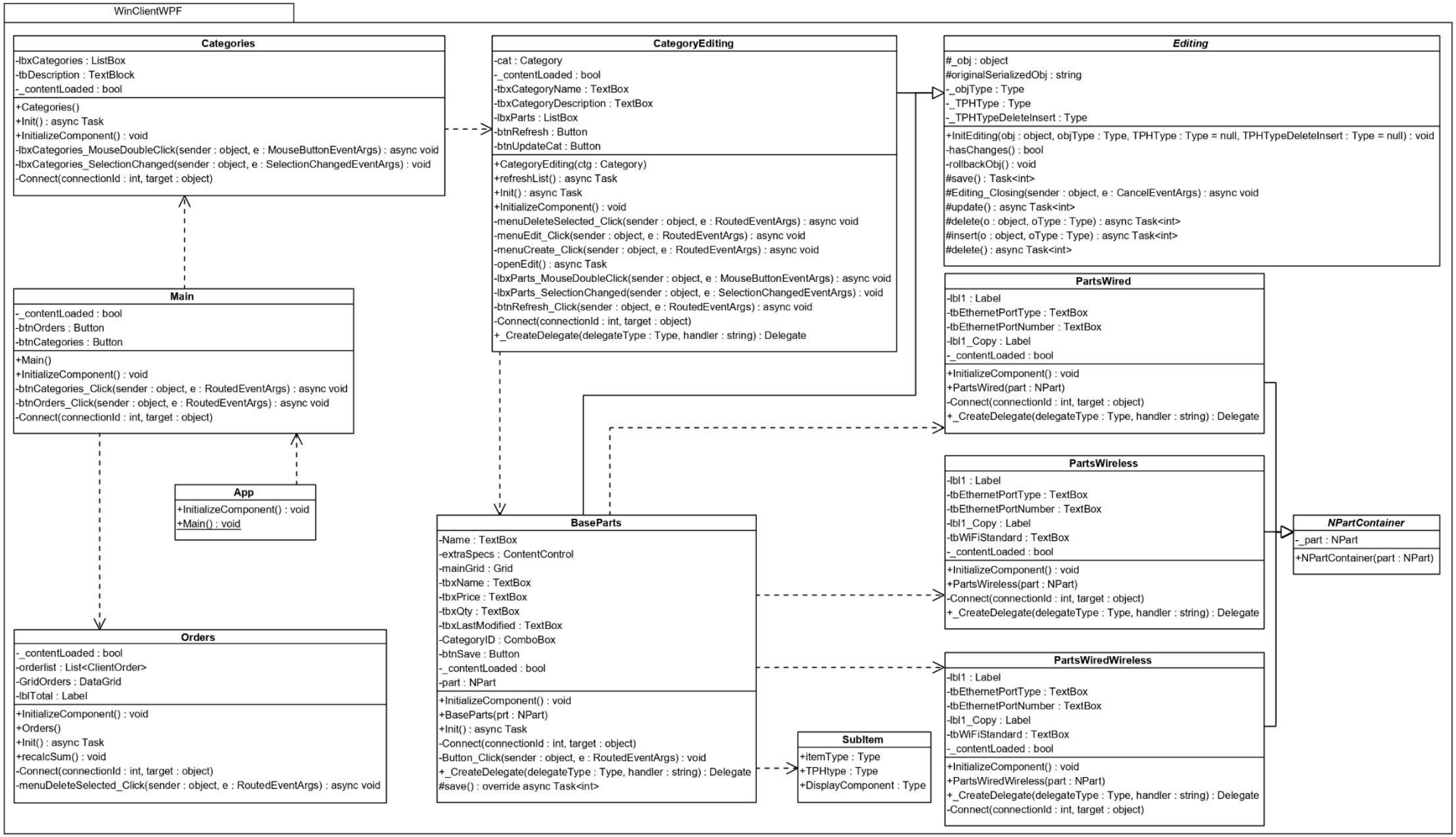
Common


Visual Paradigm Standard(Oleg Sivers(Nelson Marlborough Institute of Technology))
SDV701common
utils
+GetAssemblyByName(name : string) : Assembly
+GetAssemblyNameByType< T >() : string
+DeserializeJsonWithTypes< T >(json : string) : List<T>
+DeserializeJsonWithTypesSingle< T >(json : string) : T
+CloneObject< T >(obj : T) : T
+SerializeJsonWithTypes(obj : object) : String
Model
+IsValid() : string
Category
+id : int
+Categoryname : string
+Description : string
+ToString() : override string
+IsValid() : override string
ClientOrder
+id : int
+PartsID : int
+Quantity : int
+Price : float
+OrderDateTime : DateTime
+ClientName : string
+ClientEMail : string
+Proceeded : bool
+Name : string
+Summ : float
+Currency : string
+ToString() : override string
+IsValid() : override string
NPart
+id : int
+Name : string
+ClassName : string
+Price : float
+Currency : string
+QtyInStock : int
+CategoryID : int
+LastModified : DateTime
+ToString() : override string
+NPart()
+IsValid() : override string
+ReturnOODetails() : string
NEmptyPart
NWiredPart
+EthernetPortType : string
+NumberOfPorts : int
+IsValid() : override string
+ReturnOODetails() : override string
NWiredWirelesspart
+WiFiStandard : string
+EthernetPortType : string
+NumberOfPorts : int
+IsValid() : override string
+ReturnOODetails() : override string
NWirelesspart
+WiFiStandard : string
+IsValid() : override string
+ReturnOODetails() : override string

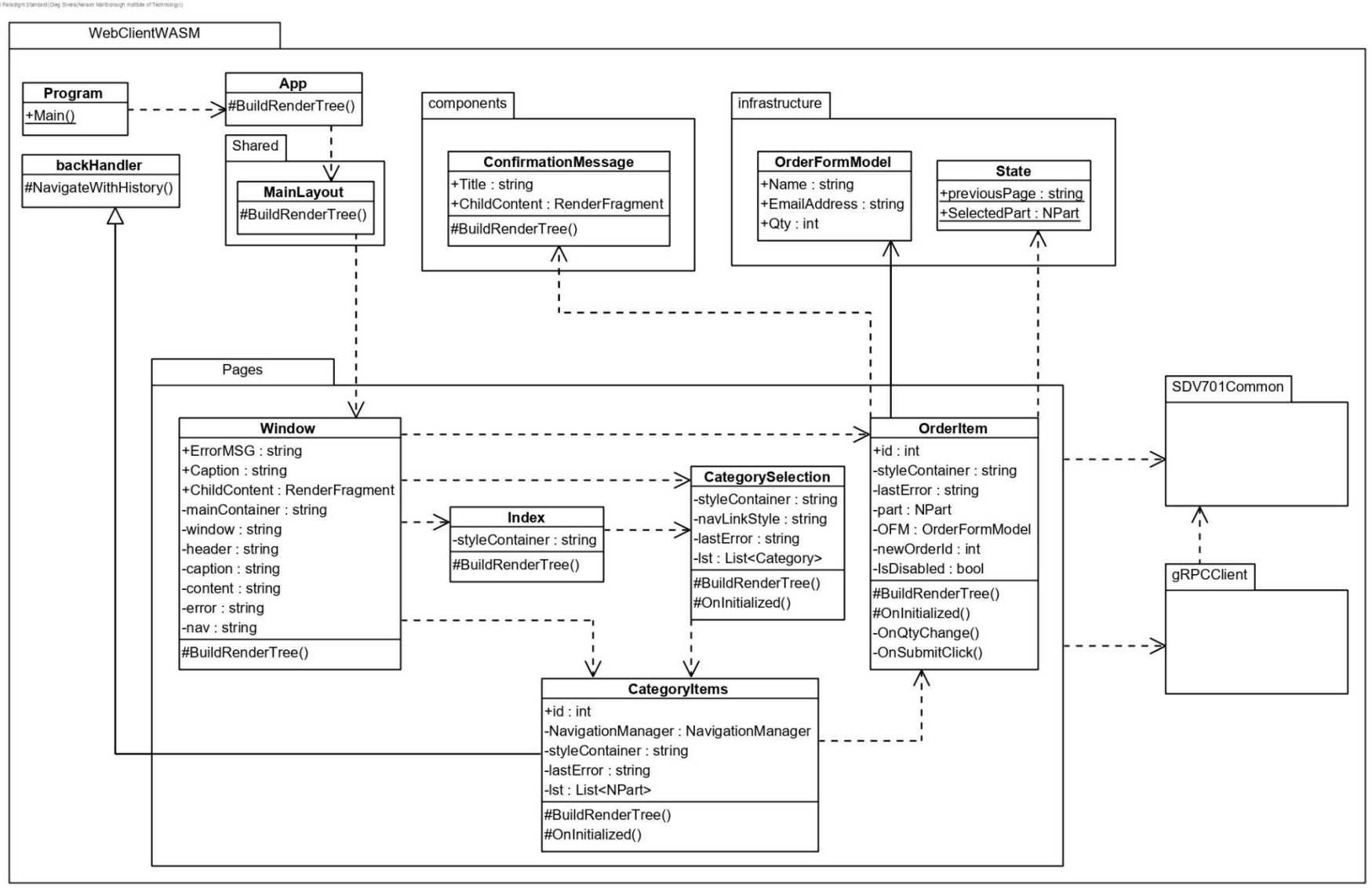
Back-end


SDV701BackEnd
utils
ConnectionSettings
+ConnectionString : string
+ProviderName : string
+DBProvider : DbProviderFactory
+InitProvider() : void
SettingsBackEnd
+CS : ConnectionSettings
SettingsManager
+config : IConfiguration
QueryGenerator
-GenerateObject(asm : Assembly, t : Type, JSON : string) : object
+Update(assemblyName : string, typeName : string, JSON : string, TPHType : string) : query
+Insert(assemblyName : string, typeName : string, JSON : string, TPHType : string) : query
+Delete(assemblyName : string, typeName : string, JSON : string, TPHType : string) : query
query
+sql : string
+paramList : ParamList
Program
+Main(args : string[]) : void
+CreateHostBuilder(args : string[]) : IHostBuilder
Startup
+ConfigureServices(services : IServiceCollection) : void
+Configure(app : IApplicationBuilder, env : IWebHostEnvironment) : void
gRPCProtos
GetByIdRequest
+Id : int
basicResponse
+ErrorResponse : string
SetRequest
+SystemType : string
+Assembly : string
+JSON : string
+TPHType : string
GetJsonSerializedResponse
+Response : string
+Error : basicResponse
DeleteUpdateResponse
+Error : basicResponse
+RowsAffected : int
PlaceOrderRequest
+PartsId : int
+PartsQty : int
+ClientName : string
+ClientEMail : string
InsertResponse
+Error : basicResponse
+InsertedID : int
GetHierarchicalJsonSerializedResponse
+SystemType : string
+Assembly : string
+Response : string
+Error : basicResponse
GetAllRequest
-_logger
Using requests and responses
DB
ParamList
TableForMapping
+Map< T >() : List<T>
+MapHierarchy< T >(types : List<Type>, FieldContainingType : string) : List<T>
DBExecuter
+SQLRequest(sql : string, paramList : ParamList) : TableForMapping
+SQLRequestDTO(sql : string, paramList : ParamList) : TableForMapping
+SQLRequestSPAutoFillParams(sql : string, paramList : ParamList) : TableForMapping
+SQLToReader(sql : string) : TableForMapping
+execNonQuerySPAutoFillParams(sql : string, paramList : ParamList) : int
+execNonQuery(sql : string, paramList : ParamList) : int
+execScalar(sql : string, paramList : ParamList) : int
+execScalarSPAutoFillParams(sql : string, paramList : ParamList) : int
ConnectionHelper
-isDisposed : bool
+connection : DbConnection
-ConstructParam(param : KeyValuePair<string, object>) : DbParameter
+ConnectionHelper(ConnectionStr : string)
+MakeCommand(sql : string, paramList : ParamList, commandType : CommandType = CommandType.Text) : DbCommand
+Dispose() : void
#Dispose(disposing : bool) : void
NetshopService
-_logger : readonly ILogger<NetshopService>
+NetshopService(logger : ILogger<NetshopService>)
+GetParts(request : GetAllRequest, context : ServerCallContext) : override Task<GetHierarchicalJsonSerializedResponse>
+GetCategories(request : GetAllRequest, context : ServerCallContext) : override Task<GetJsonSerializedResponse>
+GetOrders(request : GetAllRequest, context : ServerCallContext) : override Task<GetJsonSerializedResponse>
+GetPartsByCategoryId(request : GetByIdRequest, context : ServerCallContext) : override Task<GetJsonSerializedResponse>
+Delete(request : SetRequest, context : ServerCallContext) : override Task<DeleteUpdateResponse>
+Update(request : SetRequest, context : ServerCallContext) : override Task<DeleteUpdateResponse>
+Insert(request : SetRequest, context : ServerCallContext) : override Task<InsertResponse>
+GetPartById(request : GetByIdRequest, context : ServerCallContext) : override Task<GetJsonSerializedResponse>
+PlaceOrder(request : PlaceOrderRequest, context : ServerCallContext) : override Task<InsertResponse>
Using Model - related classes
SDV701Common

WPF Client


WinClientWPF
Categories
-lbxCategories : ListBox
-tbDescription : TextBlock
-_contentLoaded : bool
+Categories()
+Init() : async Task
+InitializeComponent() : void
-lbxCategories_MouseDoubleClick(sender : object, e : MouseButtonEventArgs) : async void
-lbxCategories_SelectionChanged(sender : object, e : SelectionChangedEventArgs) : void
-Connect(connectionId : int, target : object)
CategoryEditing
-cat : Category
-_contentLoaded : bool
-tbxCategoryName : TextBox
-tbxCategoryDescription : TextBox
-lbxParts : ListBox
-btnRefresh : Button
-btnUpdateCat : Button
+CategoryEditing(ctg : Category)
+refreshList() : async Task
+Init() : async Task
+InitializeComponent() : void
-menuDeleteSelected_Click(sender : object, e : RoutedEventArgs) : async void
-menuEdit_Click(sender : object, e : RoutedEventArgs) : async void
-menuCreate_Click(sender : object, e : RoutedEventArgs) : async void
-openEdit() : async Task
-lbxParts_MouseDoubleClick(sender : object, e : MouseButtonEventArgs) : async void
-lbxParts_SelectionChanged(sender : object, e : SelectionChangedEventArgs) : void
-btnRefresh_Click(sender : object, e : RoutedEventArgs) : async void
-Connect(connectionId : int, target : object)
+_CreateDelegate(delegateType : Type, handler : string) : Delegate
Editing
#_obj : object
#originalSerializedObj : string
-_objType : Type
-_TPHType : Type
-_TPHTypeDeleteInsert : Type
+InitEditing(obj : object, objType : Type, TPHType : Type = null, TPHTypeDeleteInsert : Type = null) : void
-hasChanges() : bool
-rollbackObj() : void
#save() : Task<int>
#Editing_Closing(sender : object, e : CancelEventArgs) : async void
#update() : async Task<int>
#delete(o : object, oType : Type) : async Task<int>
#insert(o : object, oType : Type) : async Task<int>
#delete() : async Task<int>
Main
-_contentLoaded : bool
-btnOrders : Button
-btnCategories : Button
+Main()
+InitializeComponent() : void
-btnCategories_Click(sender : object, e : RoutedEventArgs) : async void
-btnOrders_Click(sender : object, e : RoutedEventArgs) : async void
-Connect(connectionId : int, target : object)
App
+InitializeComponent() : void
+Main() : void
PartsWired
-lbl1 : Label
-tbEthernetPortType : TextBox
-tbEthernetPortNumber : TextBox
-lbl1_Copy : Label
-_contentLoaded : bool
+InitializeComponent() : void
+PartsWired(part : NPart)
-Connect(connectionId : int, target : object)
+_CreateDelegate(delegateType : Type, handler : string) : Delegate
PartsWireless
-lbl1 : Label
-tbEthernetPortType : TextBox
-tbEthernetPortNumber : TextBox
-lbl1_Copy : Label
-tbWiFiStandard : TextBox
-_contentLoaded : bool
+InitializeComponent() : void
+PartsWireless(part : NPart)
-Connect(connectionId : int, target : object)
+_CreateDelegate(delegateType : Type, handler : string) : Delegate
NPartContainer
-_part : NPart
+NPartContainer(part : NPart)
BaseParts
-Name : TextBox
-extraSpecs : ContentControl
-mainGrid : Grid
-tbxName : TextBox
-tbxPrice : TextBox
-tbxQty : TextBox
-tbxLastModified : TextBox
-CategoryID : ComboBox
-btnSave : Button
-_contentLoaded : bool
-part : NPart
+InitializeComponent() : void
+BaseParts(prt : NPart)
+Init() : async Task
-Connect(connectionId : int, target : object)
-Button_Click(sender : object, e : RoutedEventArgs) : void
+_CreateDelegate(delegateType : Type, handler : string) : Delegate
#save() : override async Task<int>
PartsWiredWireless
-lbl1 : Label
-tbEthernetPortType : TextBox
-tbEthernetPortNumber : TextBox
-lbl1_Copy : Label
-tbWiFiStandard : TextBox
-_contentLoaded : bool
+InitializeComponent() : void
+PartsWiredWireless(part : NPart)
+_CreateDelegate(delegateType : Type, handler : string) : Delegate
-Connect(connectionId : int, target : object)
SubItem
+itemType : Type
+TPHtype : Type
+DisplayComponent : Type
Orders
-_contentLoaded : bool
-orderlist : List<ClientOrder>
-GridOrders : DataGrid
-lblTotal : Label
+InitializeComponent() : void
+Orders()
+Init() : async Task
+recalcSum() : void
-Connect(connectionId : int, target : object)
-menuDeleteSelected_Click(sender : object, e : RoutedEventArgs) : async void

WEB Client


WebClientWASM
Program
+Main()
App
#BuildRenderTree()
backHandler
#NavigateWithHistory()
Shared
MainLayout
#BuildRenderTree()
components
ConfirmationMessage
+Title : string
+ChildContent : RenderFragment
#BuildRenderTree()
infrastructure
OrderFormModel
+Name : string
+EmailAddress : string
+Qty : int
State
+previousPage : string
+SelectedPart : NPart
Pages
Window
+ErrorMSG : string
+Caption : string
+ChildContent : RenderFragment
-mainContainer : string
-window : string
-header : string
-caption : string
-content : string
-error : string
-nav : string
#BuildRenderTree()
Index
-styleContainer : string
#BuildRenderTree()
CategorySelection
-styleContainer : string
-navLinkStyle : string
-lastError : string
-lst : List<Category>
#BuildRenderTree()
#OnInitialized()
CategoryItems
+id : int
-NavigationManager : NavigationManager
-styleContainer : string
-lastError : string
-lst : List<NPart>
#BuildRenderTree()
#OnInitialized()
OrderItem
+id : int
-styleContainer : string
-lastError : string
-part : NPart
-OFM : OrderFormModel
-newOrderId : int
-IsDisabled : bool
#BuildRenderTree()
#OnInitialized()
-OnQtyChange()
-OnSubmitClick()
SDV701Common
gRPCClient