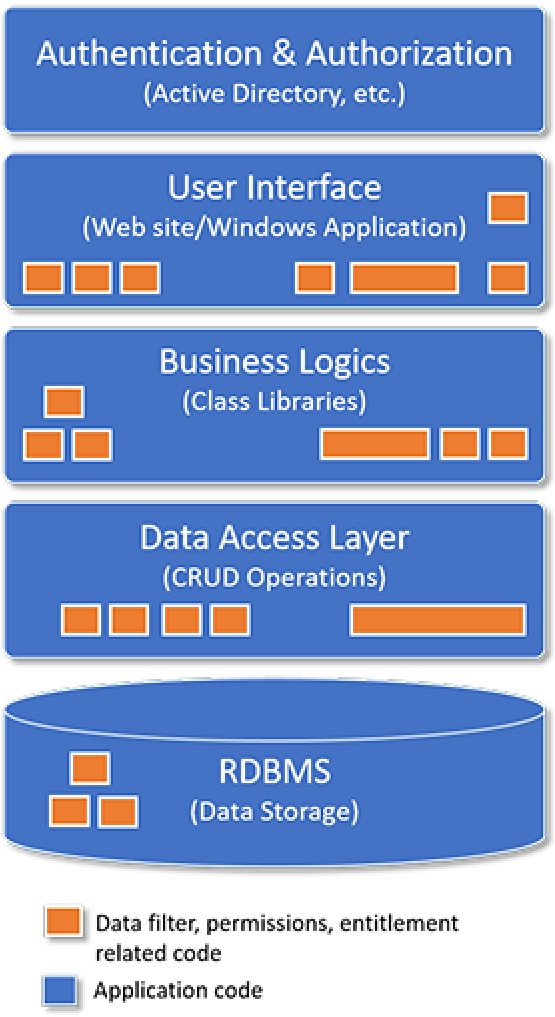
Background & Thoughts



Technology solution is vast these days, and there is always at least one solution for every technical problem. A typical application has following layers:

1. Security Layer (Authentication & Authorization)
2. User Interface Layer
3. Business Logic Layer
4. Data Access Layer
5. And a RDBMS database

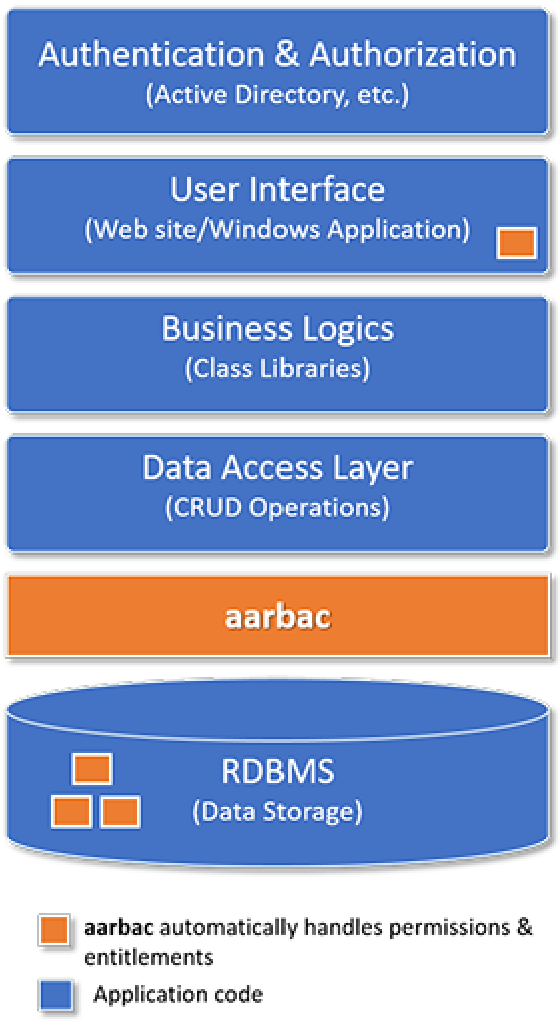
Security layer generally just performs authentication & authorization, and to facilitate roles (permissions & entitlements), developers implement various `HasPermission(), IsInGroup(), etc.` type methods with many `ifs elses & switches`, various sql queries and inject code into these layers(2, 3 & 4). Basically to enure that **users with appropriate rights are performing appropriate operations.** Users should **see** what they should see on the screens, users able to **do** what they are allowed to do with data.

During initial few releases, everything looks good, but messed up soon when complex business scenarios, exceptions are introduced, more `ifs, switches` are introduced in those layers, code quality decreases and code volume increases and makes production support & enhancements difficult. Nightmare for new team members!

What is aarbac?

aarbac is **An Automated Role Based Access Control** .NET framework which can override all CRUD(Create, Read Update, Delete) operations automatically based on the logged in user role. It separates out permission related code into a complete new layer and let these layers (2, 3 & 4) do their regular job & not worried about the permission at all. Additionally it also maintains user entitlements.

1. Automated - Data Filter & Permissions are abstracted into a separate layer and all automated.
2. Schema based - Data filters and permissions are based on your database schema.
3. Clean Code - Clean code, less error, less testing, less maintenance.
4. Encrypted - The role, entitlements & user parameters are encrypted and stored as binary in the aarbac database.
5. Disable overriding for specific queries - As we do understand there will be few queries where aarbac may not able to produce automated result as desired, in those cases, just switch off aarbac.
6. Comes with REST API, Utility, WinApp testbed, Sample Code.
7. Pluggable.
8. nuget deployed
9.



1. Automated Data Filters & Permissions (Schema Based)

Apply row & column level permissions on your SELECT,INSERT,UPDATE & DELETE queries. For example, a read (or select) operation like the following …

```
select * from Author
```

automatically may get converted to...

```
SELECT Author.AuthorId, Author.Name, Author.ZipCodeId FROM Author
inner join [ZipCode] [t9] on [t9].ZipCodeId = [Author].ZipCodeId
inner join [City] [t10] on [t10].CityId = [t9].CityId WHERE t10.Name in ('New
York','Charlotte')
```

...assuming user belongs to a role which allows him to see only 3 columns from author table and only allowed to see authors from New York and Charlotte cities.

And an update query like the following...

```
update Author set Name = 'Eyedia', SSN = '999-99-9999' where AuthorId = 9999
```

may hit exception like...

```
- User 'abc' does have permission to update table 'Author' but does not have permission
to update column 'SSN'
```

Sample Code

Select:

```
using (Rbac rbac = new Rbac("essie"))   //<-- you should pass the logged in user name
from the context
{
    using (RbacSqlQueryEngine engine = new RbacSqlQueryEngine(rbac, query))
    {
        engine.Execute(); //<-- automatically parse and transform query based on role
        if ((!engine.IsErrored) && (engine.Parser.IsParsed)
            && (engine.Parser.QueryType == RbacQueryTypes.Select))
            return engine.Table; //<-- if it is select query, the table will be loaded
    }
}
```

Inserts, updates and deletes

```
using (Rbac rbac = new Rbac("essie"))   //<-- you should pass the logged in user name
from the context
{
    using (SqlQueryParser parser = new SqlQueryParser(rbac))
    {
        parser.Parse(query); //<-- this will throw exception if not permitted
        //<-- if you are here, you are goood. Just perform basic insert/update/delete
    }
}
```

---

2. Entitlements (Menu/Sub-Menu & Screen/Screen-Elements)

Every rule in aarbac has screen entitlement, you can define entitlements for your applications in following 2 categories:

1. Menus - Menu and sub menues within (linked list nodes)

2. Screens - Screen and Screen Elements (linked list nodes)

And just set visible and enabled properties on those nodes.

When user logs in, i.e. authenticated and authorized based on your authentication mechanism(for example active directory of the organization), just map user with a specific aarbac role, & each role will have entitlement. App developers need to apply the entitlement xml into the menu and screen elements.