.Net LevelUp

Intro

What we will do

LevelUp consists of a series of tasks that walk you through the .NET web applications development with touching of some frontend. The goal is to get as much experience as possible during task implementation so the majority of internship is practice. After successful task passing mentees will be able to contact their mentors for any questions they have and this is how it's going to work.

The program stack is .NET Core, C#, PostgreSQL, EF Core, ASP.NET Core, xUnit, Git, Angular which is today's robust stack for web-projects of any complexity.

Follow the principles below to be successful and efficient in proceeding with tasks

- Always pass your best code, refactor anything you consider worth it and, again, pass your best code.
- Your commited code should always work, so please don't commit code with build errors/runtime exceptions.
- Use Git during your development, it helps you to execute tasks partially, commit every small piece you've done: You implemented a new class/service/repository/controller? Do the commit! Rewrote mapping, added helper, reworked DI configuration? Do the commit!
- https://docs.microsoft.com/en-us/dotnet is the best documentation for .NET you'll ever find.

Web app

Web app requirements

Clothing online store. You can reference asos.com or urbanoutfitters.com as a sample for categories, products, etc.

- Two main sections of the application: women's clothing, men's clothing.
- System of categories and subcategories, for example shoes -> sandals. In different sections there are different categories and subcategories, respectively, as an example, only women have a section for dresses and skirts. However some of the categories can be shared for example shoes -> sandals is a common category for both women and men sections.
- Product brand - common for all sections. Should be unique across the platform.
- Product itself.
- Support for submitting an order. Order has a couple of statuses (for example: In Review, In Delivery, Completed) where Completed is the final one.
- Ability to leave a review with a rating and a comment.
- Customer's account with the ability to add personal data (email, phone) and delivery address.
- Admin panel for managing the store.

Domain entities and operations on it

- Section (women, men)
  - Add new section
  - Update existing section (rename)
- Brand
  - Add new brand
  - Update brand
  - Assign/unassign product to a bran
- Cart
  - Add/remove item from cart

Domain entities and operations on it

- Category (subcategories as well)
  - Add new category / subcategory
  - Rename category/subcategory
  - Link category to existing section
  - Link subcategory to existing category
  - Consider the case when a subcategory is linked to a category but should not be shown for all sections. For example shoes => sneakers are common for men/women sections but shoes => high heels are only for women.

Domain entities and operations on it

- Product (price, quantity, description, image)
  - Add new product
  - Update product
  - Check product availability
  - Quantity is limited (product was sold out => available quantity is 0 => out of stock)
- User (with phone and address if customer)
  - Register new user
  - Login as a new user
  - Update user personal information

Domain entities and operations on it

- Order
  - Create order for customer
  - Add/Remove product to an order
  - Submit order
  - Support order status transition. For example: In Review => In Delivery => Completed)
- Review (rating + comment)
  - Add new review with star rating and comment

Git

Study objective

Learn how to work with the git version control system.

Home work (Task)

Learn how to work with the git version control system. Theory and Task:

- Install git. English version only!
- Complete https://githowto.com
- Complete https://learngitbranching.js.org
- Read about git, try out the most essential commands.
- Understand how it can help you in development. If you wouldn't use it tomorrow in the next task - then please re-read everything in this stage again!

Home work (Questions)

- How does git help even a solo developer?
- How does git help two or more developers work on the same project?
- When would you store changes in two (or more) commits? When in the one? Why?
- How to stage only several files (not all not staged)?
- Why do conflicts happen?
- Explain the mechanism of solving conflicts.
- What is the difference between merge and rebase?
- When would you stash your changes?
- What are possible cases of using force push?
- What is the reset command? What are different types of reset? (hard, mixed, soft)

SQL Overview

Study objective

Learn basic SQL concepts. Learn how to build relations between entities. Learn what normalization is.

Theory

- https://www.ibm.com/cloud/learn/relational-databases
- https://www.mongodb.com/nosql-explained
- https://www.sqlshack.com/learn-sql-types-of-relations
- https://docs.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description
- http://svyatoslav.biz/relational_databases_book_download_typographic

Home work (Task)

You can use any development tool for ERD. Diagram should include all the entities for the e-store, all properties with data types and relations between tables.

- Read about relational, NoSQL databases.
- Read about relations between tables in relational databases.
- Read about database normalization.
- Implement Entity Relationship Diagram for your future web application.

Home work (Questions)

- What is the difference between relational and NOSQL databases? Where are they used? When SQL does suit better than NoSQL, and otherwise?
- How to implement one-to-one, one-to-many and many-to-many relationships? What is Primary Key? What is Foreign Key? What other unique constraints are available?
- What are 1NF, 2NF and 3NF? Why is it important to have your database normalized?
- What are the cases to denormalize your database?

DB model with SQL

Study objective

Learn creating database models with SQL operations, PostgreSQL data types and constraints.

Theory

- https://www.postgresql.org/docs/current/ddl.html
- https://www.postgresql.org/docs/current/dml.html
- https://www.postgresql.org/docs/current/datatype.html
- https://www.postgresql.org/docs/current/indexes-intro.html
- https://habr.com/ru/post/247373
- https://habr.com/ru/company/postgrespro/blog/326096

Home work (Task)

As a result you should have a set of sql files that allows you to recreate database structure.

- We will be using PostgreSQL as a default sql provider.
- Convert your ERD diagram into a set of SQL scripts to create database structure from scratch. Specify all tables, columns, data types and constraints for each table. Also specify ON UPDATE, ON DELETE constraints.

Home work (Questions)

- What are the methods to create/delete database in PostgreSQL?
- How to implement one-to-one, one-to-many and many-to-many relationships? What is Primary Key? What is Foreign Key? What other unique constraints are available?
- What is an index? When would you use it? What is the difference between a Clustered and Non Clustered Index?

Aggregations with SQL

Study objective

Learn aggregation using SQL operations: SELECT, WHERE, JOIN, GROUP BY, ORDER BY, etc.

Theory

- https://www.postgresql.org/docs/current/queries.html
- https://habr.com/ru/post/480838
- https://habr.com/ru/post/47031 (for MySQL)
- https://www.postgresqltutorial.com/postgresql-views

Home work (Task)

Implement list of functions to aggregate over domain:

- Get all products of selected brand
- Get all product variations for the selected product
- Select all brands with the number of their products respectively. Order by the number of products.
- Get all products for a given category and section.
- Get all completed orders with a given product. Order from newest to latest.
- Get all reviews for a given product. Implement this as a viewtable which contains rating, comment and info of a person who left a comment.

Home work (Questions)

- How does SELECT work in technical terms? What are the main query process steps?
- What is the execution plan and how does it help to profile your queries?
- What is the full table scan? How to detect and avoid full table scans?
- How does JOIN work? What are the different types of JOIN command? What are the best practices when working with JOIN?
- How does ordering and grouping work?
- What is the view in SQL? What are the use cases to work with view instead of the actual table?

EF Core overview. Model migrations

Study objective

Learn what ORM is and how it works. Learn Basic EF Core functionality. Learn how to embed EF Core in any .NET application. Learn about DB-first and Code-first approaches. Learn about using Fluent API to build entities and relations

Theory

- https://docs.microsoft.com/en-us/ef/core

Home work (Task)

All tasks should be implemented with simple console applications, no web apps required.

- Learn basic concepts around ORMs and EF Core in particular.
- Setup application with EF Core configuration.
- Try code first approach. Scaffold database to generate domain entities and DB Context.
- Create entities and configure relations using Fluent API (each configuration must be in a separate file).
- Try database first approach. Create migrations and generate DB based on Fluent API configuration.
- As a result you should have a console application with configured EF Core setup, list of models and migrations to follow code-first implementation.

Home work (Questions)

- What is ORM and how does it work? What are available instruments to work with a database in .NET?
- What is ADO.NET technology? What is LinqToSql and what is EF? What is the Dapper library?
- What are the main classes EF Core provides to work with DB?
- What are two main ways of describing models and relations in EF Core? What are the main differences between them?
- What is the navigation property in terms of EF? What are the naming conventions when using navigation properties?
- How to configure different types of relationships using EF? (1to1, 1toM, MtoM)
- Why would you need to declare your navigation properties as virtual? What is the technical explanation behind this?
- What is migration and why would you need it?
- What are the advantages and disadvantages of code first approach over the database first?

Aggregations with EF Core

Study objective

Learn how to use EF Core to aggregate over databases. Practice LINQ as part of work with EF.

Theory

- https://docs.microsoft.com/en-us/ef/core/querying
- https://docs.microsoft.com/en-us/ef/core/change-tracking
- https://docs.microsoft.com/en-us/ef/core/change-tracking/explicit-tracking
- https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx
- https://stackoverflow.com/questions/54274166/how-does-ef-core-modified-entity-state-behave

Home work (Task)

- Build queries on top of your code-first console application.
- Implement the same 6 queries from Aggregations with SQL stage but querying DB with EF Core. Ignore view table requirement for last function.
- All queries should be executed in two mods: Lazy loading and Eager loading. Be ready to show generated by EF sql script and explain the difference.
- Execute one update request with disconnected scenario using Explicit Tracking and Entity State update.
- No need to create a menu, you can call them all on startup with hardcoded arguments

Starting from now we use our code-first project as a basis for all future work. Later it will be converted to a web application that we will improve on each stage.

Home work (Questions)

- What are different approaches to load related data in EF Core? What are the use-cases for all of them?
- What is client vs server evaluation? What are the common cases when EF performs evaluations on the client side and what problems can client evaluation bring?
- What are tracking and no-tracking queries? What is the reason behind tracking queries? What are the use-cases not to track entities? How to track entities explicitly from code?
- What is the N+1 query problem and how would you solve it?
- What is the “Cartesian Explosion” problem and how would you solve it?
- Why would you need to declare your navigation properties as virtual? What is the technical explanation behind this?
- What are connected and disconnected scenarios? What is the use-case for disconnected scenarios and how to work with it properly?

Web architecture overview

Study objective

Learn different approaches about how to design web applications. Learn common concepts about how to organize client-server interaction with HTTP. Learn best practices for building microservices.

Theory

- https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles
- https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures
- https://restfulapi.net/
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://www.c-sharpcorner.com/article/microservice-using-asp-net-core/
- https://microservices.io/patterns/microservices.html (very useful, save for future reference)
- https://docs.microsoft.com/en-us/azure/architecture/microservices/ (save for future)
- https://docs.microsoft.com/en-us/previous-versions/msp-n-p/dn568099(v=pandp.10) (save for future)
- https://habr.com/ru/articles/543828/
- https://habr.com/ru/post/427739/
- https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice
- https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns

Home work (Task)

- Learn about Onion, Layered and DDD service architectures.
- Learn about how REST principles help to build service API.
- Learn about CQRS usage in web applications.
- Learn about microservices.

In our project we will go with the Layered architecture but it’s important to understand different approaches and situations where you would like to use them

Home work (Questions)

- What is the point of splitting your code into different layers? How does dependency inversion help?
- Please describe the main principles of N-layer architecture.
- Please describe the main principles of Onion(Clean) architecture.
- What are the main differences between N-layer and onion architectures?
- Please describe the main principles of DDD. What are the main terms and concepts that are used in DDD?
- How does CQRS help to build loosely coupled layers and applications?
- What is REST API? What are the main principles behind this concept? How does this help to build robust web applications? What is the connection between REST and HTTP?
- What is microservice architecture? What are the pros and cons compared to monolithic architecture?

ASP.NET Core

Study objective

Learn how to create web applications using ASP.NET Core. Learn about controllers, routes, http attributes. Learn about model validation in ASP.NET Core.

Theory

- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-7.0&tabs=windows
- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/mvc/models/model-binding?view=aspnetcore-7.0

Home work (Task)

- Rework your console application into a web application. Include ASP.NET Core package and setup startup configuration.
- Create your first controller that will be responsible for brands of your clothing store.
- Create a RESTful API to create/update/get/delete brands.
- Don’t forget about input model validation (use FluentValidation) and using attributes for routing and http configuration.
- For now you can have all database operations within this controller.

Here we start our web application on top of previously created console app. We will start with a single controller and later expand this implementation.

Home work (Questions)

- What is ASP.NET Core and ASP NET? What are the benefits of using ASP.NET Core over ASP.NET?
- What is the responsibility of Startup class in ASP.NET Core? What is the difference between Configure and ConfigureServices methods?
- What is ASP.NET Core middleware? What is Http request pipeline?
- How Configuration works in ASP.NET Core? What is stored in Appsettings.json file. What is Options pattern?
- Please describe the role of the controllers in ASP.NET Core. What is the difference between Controller and ControllerBase classes? Why is it important to derive from ControllerBase class when writing a controller?
- What is an endpoint in terms of web application?
- What is routing, and how can you define routes in ASP.NET Core?
- What are the differences between conventional and attribute-based routings? What is the preferred way to define routes in ASP.NET Core?
- What is model binding in ASP.NET Core? What is the role of parameter-based attributes (FromBody, FromQuery, etc). When would you use each of them?
- Please explain model validation process. What is ModelState class? What are validation attributes?

App architecture. DI

Study objective

Learn how to design your web application using the Layered architecture. Learn how DI works and different types of lifetime.

Theory

- https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-7.0
- https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection
- https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection-guidelines
- https://martinfowler.com/articles/injection.html

Home work (Task)

- Split your brands controller logic into layers according to Layered architecture principles.
- Configure DI for your web application. Use default DI implementation provided by ASP.NET Core framework. Don’t forget about the lifetime!

Here we achieve well-organized architecture where we can easily add new controllers, services, repositories, etc.. So at this stage you will have a full-fledged web application that allows you to work with one of the entities of your domain.

Home work (Questions)

- What is dependency injection? How would this help us when building web app?
- What is Service Locator Pattern and what is the difference between this and DI? Why is service locator considered as an anti-pattern?
- What is the lifetime in terms of DI? What are the options when using ASP.NET Core DI extension? Could you please provide examples when each lifetime might be useful?
- What are other well-known DI libraries for .NET Core applications? What are their advantages and disadvantages compared to standard ASP.NET Core DI extension?

API design and implementation

Study objective

Learn how to compose different parts of your application and how to reuse existing logic.

Theory

- https://www.scaleyourapp.com/what-is-swagger-and-why-do-you-need-it-for-your-project/
- https://swagger.io/docs/specification/about/
- https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-7.0&tabs=visual-studio
- https://learning.postman.com/docs/getting-started/introduction/
- https://apiguide.readthedocs.io/en/latest/build_and_publish/use_RESTful_urls.html

Home work (Task)

- Design and implement your e-store application as a set of controllers that are responsible for different business actions.
- Think a lot about what actions make sense for your domain, please don’t make CRUDs for every entity even when it’s not needed. Think about the real world actions and try to project them into the set of controllers and actions. Refer to requirements.
- Application should be designed according to the Layred architecture principles, reuse logic where you can, validate the state of your application. Design endpoints according to REST principles.
- Document your APIs using Swagger (Swashbuckle library). Decorate all you endpoints with returned types and status codes. Take a look at ProducesResponseType attribute
- Use Postman for API testing

By the end of this stage we have a robust web application.

Home work (Questions)

- What is Postman? What are other alternatives to reach your API endpoints? Is it possible to test the API using a browser?
- What is Swagger? How does documentation of our API help in development?

Unit and integration testing

Study objective

Learn what unit tests and integration tests are. Learn how to mock dependencies and test isolated modules. Learn how tosetup integration tests.

Theory

- https://www.geeksforgeeks.org/types-software-testing/
- https://docs.microsoft.com/en-us/learn/modules/visual-studio-test-concepts/
- https://stackify.com/asp-net-core-testing-tools/
- https://docs.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-7.0

Home work (Task)

- Learn about the current robust stack for unit-testing: XUnit + EF Core In-Memory Database + NSubstitute + FluentAssertions.
- Write at least 10 unit-tests for any service. Half of them should cover get functionality, another half is for create/update operations.
- Learn about how to write integration tests using nUnit/xUnit.

Home work (Questions)

- What are the different types of tests and what are theirpurposes?
- What parts/layers of the application are worth covering with unit tests?
- What are the common cases for integration tests? What challenges will you face with integration tests (setup/cleanup)?

Authorization in ASP.NET Core

Study objective

Learn about how to setup authorization in ASP.NET Core. Learn what a JWT token is and how to setup Bearer scheme authorization in ASP.NET Core.

Theory

- https://docs.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-7.0
- https://docs.microsoft.com/en-us/aspnet/core/security/authorization/introduction?view=aspnetcore-7.0
- https://codepedia.info/jwt-authentication-in-aspnet-core-web-api-token
- https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity

Home work (Task)

- Learn about how authorization is configured in ASP.NET Core.
- Learn about JWT token
- Implement JWT authorization in our web application. Authorization is simple, 2 roles: admin and customer. Authorization is based on email and password.
- Decorate all controllers/endpoints with authorize attribute.
- For login and registration create a html view with a couple of inputs to submit data. Use Angular and Angular Material component library.

Home work (Questions)

- What is authentication and authorization?
- What is JWT token, what are the main components of this token?
- Explain JWT flow:
  - What access and refresh tokens are responsible for?
  - Can a server have only access tokens without refresh ones? If yes what’s pros and cons of such solution?
  - What pros and cons of storing access/refresh tokens in a database compared to not storing it?
  - How does a modern web application understand a client is unauthorized?

Final Task

Overview

This is the final task which tests your skills in planning, decomposing, learning and implementing new technologies under conditions of limited time. Communication, clarification of requirements and thinking about different nuances and details are also very important. The work will be assessed comprehensively, 100% completion of all tasks is not the only acceptable result.

Rules and definitions

- We introduce three main terms: in stock quantity (total amount of products), reserved quantity (what's already added to a shopping cart) and available quantity (what's left from in stock after all reservations), so Available = In Stock - Reserved
- Purchase process: add to cart -> submit an order
- You can't submit an order with quantity more than in stock quantity
- You can't add product to a shopping cart with quantity more than available quantity
- Technologies and libraries are up to you, but I expect websockets to be used to solve these tasks and the best way to demo your work is to have a simple UI that shows all described scenarios. You can extend your UI solution from Authorization in ASP.NET Core
- SignalR and Hangfire are recommended technologies

Home work (Task)

- When you open a certain product you should see how many users are viewing the same product in real time.
- Introduce a shopping cart so when the user puts the product in the cart, he automatically reserves it for a certain time (should be configurable). Other users in real time should see how many units of the product are reserved. If after the reservation there are no available products left, then it is impossible to add this product to the cart.
- When product quantity is changed (submitted order, canceled order, manual admin update) everyone viewing this product or having this product in a shopping cart should have updated quantity in real time.