CheatSheet für den 3. C#-Test am 24.03.2025

Table of Contents

1. Backend

1. Linq

1.1. Join

```
var joined = book // your starting point - table in the "from" statement
             .Join(author, // the source table of the inner join
                b => b.AuthorId, // select the primary key
                a => a.Id, // select the foreign key
                (b, a) => new { Book = b, Author = a }) // selection
```

2. Database

2.1. DataContext

```
public class DataContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        // add data source
        optionsBuilder.UseSqlite($"Data Source={DbPath}");
    }

    // add db entities
    public DbSet<Author> Authors { get; set; }
    public DbSet<Book> Books { get; set; }
}
```

2.2. Access

Note: Einmal async, immer async!

2.2.1. Using

```
using (var dbContext = new MyDbContext()) {
    // do something here
}
```

2.2.2. Get

```
var author = dbContext.Authors.FirstOrDefault(a => a.Id == 1);
var books = await dbContext.Books.ToListAsync();
```

2.2.3. Add

```
dbContext.Authors.Add(new Author { Name = "John Doe" });
dbContext.Books.AddRange(books);
```

2.2.4. Update

```
author.Name = "Jane Doe";
dbContext.Authors.Update(author);
dbContext.Books.UpdateRange(books);
```

2.2.5. Delete

```
dbContext.Authors.Remove(author);
dbContext.Books.RemoveRange(books);
```

2.2.6. Save Changes

```
await dbContext.SaveChangesAsync();
```

2.2.7. Include

```
dbContext.Authors
    .Include(a => a.Books)
    .ThenInclude(b => b.Genre);
```

2.2.8. Recreate Database

```
await dbContext.Database.EnsureDeletedAsync(); // delete
await dbContext.Database.EnsureCreatedAsync(); // create
```

2.2.9. Group By

```
var books = await dbContext.Books
    .GroupBy(b => b.Genre)
    .Select(g => new { Genre = g.Key, Count = g.Count() })
```

```
    .ToListAsync();
```

2.2.10. Distinct

```
var genres = await dbContext.Books
    .Select(b => b.Genre)
    .Distinct()
    .ToListAsync();
```

2.3. Include Properties

```
var allEntities = await _uow.ShippingCompanyRepository.GetAsync(
    includeProperties: new[] { nameof(ShippingCompany.CruiseShips),
$"{nameof(ShippingCompany.CruiseShips)}.{nameof(CruiseShip.ShipNames)}" }
);
```

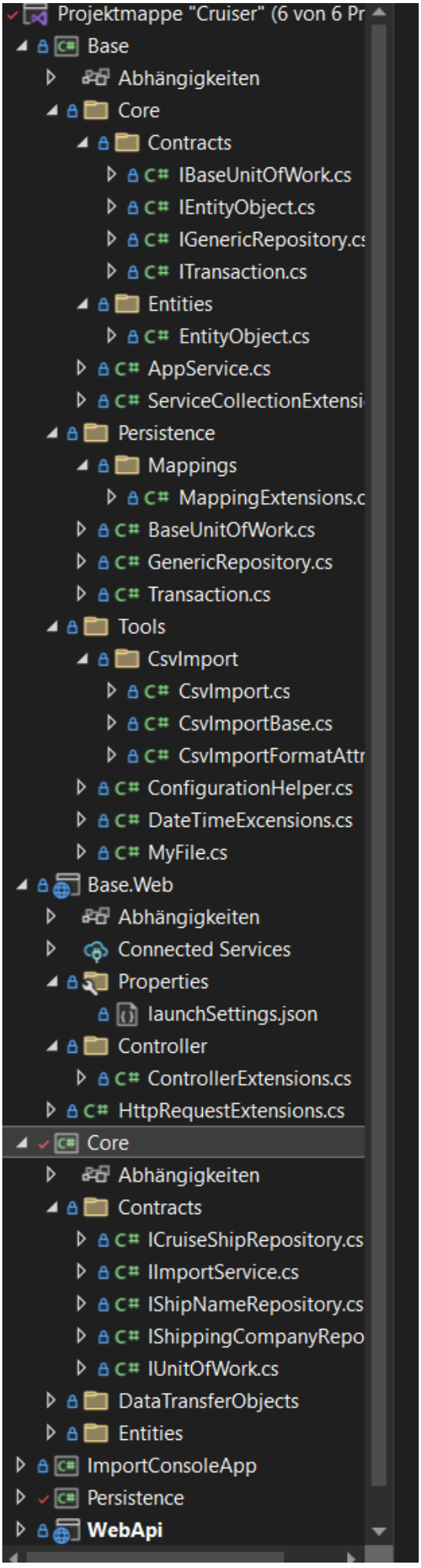
3. CSV-Import

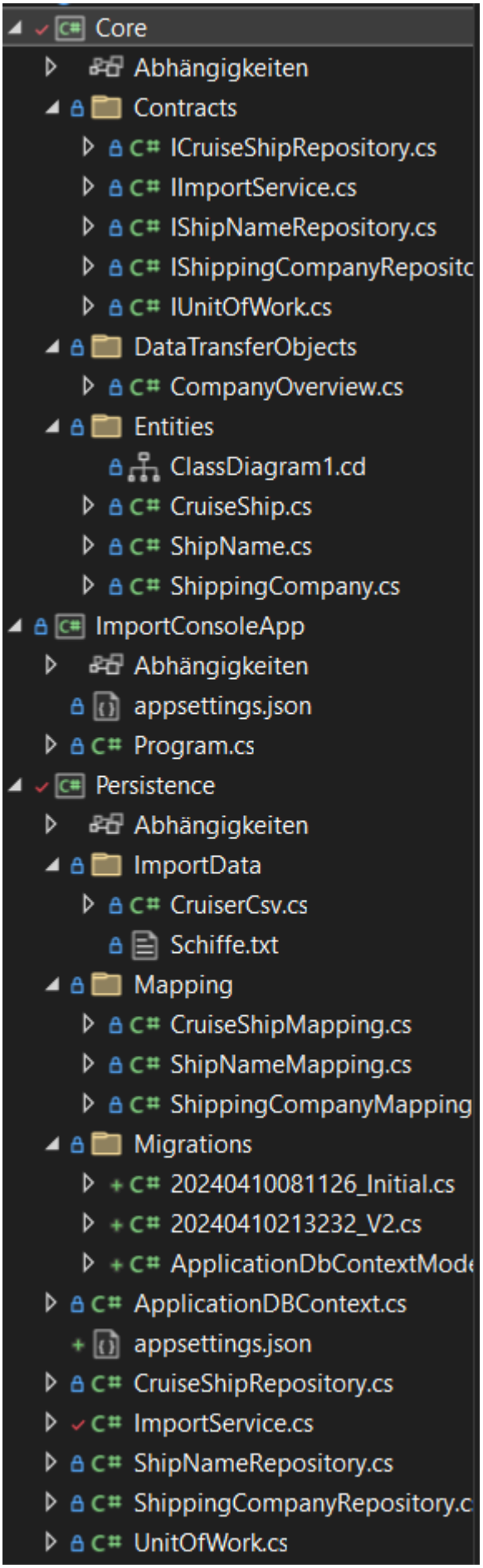
3.1. Usage

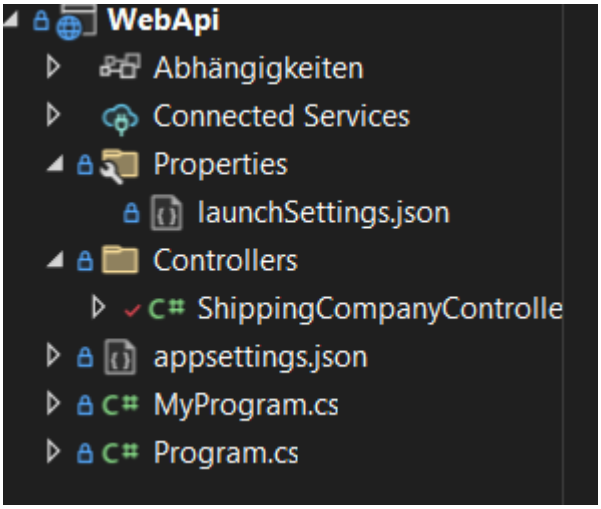
```
var productsCsv = await new CsvImport<CsvBook>()
{
    DateFormat = "yyyy/MM/dd",
    TimeFormat = "HH:mm:ss",
    Encoding   = Encoding.Unicode
}.ReadAsync(fileNameCsv);
```

4. Unit Of Work Schema

4.1. Files


Projektmappe "Cruiser" (6 von 6 Pr
Base
Abhängigkeiten
Core
Contracts
IBaseUnitOfWork.cs
IEntityObject.cs
IGenericRepository.cs
ITransaction.cs
Entities
EntityObject.cs
AppService.cs
ServiceCollectionExtensi
Persistence
Mappings
MappingExtensions.c
BaseUnitOfWork.cs
GenericRepository.cs
Transaction.cs
Tools
CsvImport
CsvImport.cs
CsvImportBase.cs
CsvImportFormatAttr
ConfigurationHelper.cs
DateTimeExcensions.cs
MyFile.cs
Base.Web
Abhängigkeiten
Connected Services
Properties
launchSettings.json
Controller
ControllerExtensions.cs
HttpRequestExtensions.cs
Core
Abhängigkeiten
Contracts
ICruiseShipRepository.cs
IImportService.cs
IShipNameRepository.cs
IShippingCompanyRepo
IUnitOfWork.cs
DataTransferObjects
Entities
ImportConsoleApp
Persistence
WebApi


Core
Abhängigkeiten
Contracts
ICruiseShipRepository.cs
IImportService.cs
IShipNameRepository.cs
IShippingCompanyReposito
IUnitOfWork.cs
DataTransferObjects
CompanyOverview.cs
Entities
ClassDiagram1.cd
CruiseShip.cs
ShipName.cs
ShippingCompany.cs
ImportConsoleApp
Abhängigkeiten
appsettings.json
Program.cs
Persistence
Abhängigkeiten
ImportData
CruiserCsv.cs
Schiffe.txt
Mapping
CruiseShipMapping.cs
ShipNameMapping.cs
ShippingCompanyMapping
Migrations
20240410081126_Initial.cs
20240410213232_V2.cs
ApplicationDbContextMode
ApplicationDBContext.cs
appsettings.json
CruiseShipRepository.cs
ImportService.cs
ShipNameRepository.cs
ShippingCompanyRepository.c
UnitOfWork.cs



4.2. Referenzen

- Core → Base
- Persistence → Core
- UnitTests → Persistence & ViewModels
- Import → Persistence
- Reports → Persistence
- Web API → Persistence
- Wpf → Persistence & ViewModels
- ViewModels → Core & Base.Wpf

5. Migration

Package Manager Console:

```
Add-Migration V1
Update-Database
```

6. DI

```
var configuration = ConfigurationHelper.GetConfiguration();

    var connectionString = configuration.GetConnectionString("DefaultConnection") ??
throw new InvalidOperationException("Connection string 'DefaultConnection' not
found.");

    AppService.ServiceCollection = new ServiceCollection();
    AppService.ServiceCollection
        .AddSingleton(configuration)
        .AddDbContext<ApplicationDbContext>(options =>
            options.UseSqlServer(connectionString))
```

```
        .AddScoped<IUnitOfWork, UnitOfWork>()
        .AddTransient<ICruiseShipRepository, CruiseShipRepository>()
        .AddTransient<IShippingCompanyRepository, ShippingCompanyRepository>()
        .AddTransient<IShipNameRepository, ShipNameRepository>()
        .AddTransient<IImportService, ImportService>();

    AppService.BuildServiceProvider();
```

7. Web API

Path:

```
app.MapGet("/api/users/{userId}", (int userId) => $"User ID: {userId}");
```

Query:

```
app.MapGet("/api/products", (string category, int page) => $"Category: {category},
Page: {page}");
```

Optional:

```
app.MapGet("/api/greet/{name?}", (string name = "Kreni") => $"Hello, {name}!");
```

Body:

```
app.MapPost("/api/create", (MyModel model) => $"Created: {model.Name}");
```

Other:

```
[HttpGet("overview")]
public async Task<ActionResult<IOrderedEnumerable<CompanyOverview>>>
GetCompanyOverview()
{
    var allEntities = await _uow.ShippingCompanyRepository.GetAsync(
        includeProperties: new[] { nameof(ShippingCompany.CruiseShips) }
    );

    var dtos = allEntities.Select(e =>
        new CompanyOverview(e.Id, e.Name, (uint)e.CruiseShips.Count)).ToList();

    dtos.Add(
        new CompanyOverview(null, "<No Company>",
            (uint)await _uow.CruiseShipRepository.CountAsync(cs => cs.ShippingCompany
== null))
```

```
    );

    return await this.NotFoundOrOk(dtos.OrderBy(c => c.CompanyName));
}
```

7.1. Delete & Update Endpoints

```
[HttpPut("{id:int}")]
public async Task<ActionResult> Update(int id, [FromBody] StationDto value)
{
    if (id.CompareTo(value.Id) != 0)
    {
        return BadRequest("Mismatch between id and dto.Id");
    }

    using (var trans = _uow.BeginTransaction())
    {
        var entity = await _uow.StationRepository.GetByIdAsync(id);
        if (entity is null)
        {
            return NotFound();
        }

        entity.Name        = value.Name;
        entity.Code        = value.Code;
        entity.Type        = value.Type;
        entity.StateCode   = value.StateCode;
        entity.IsRegional  = value.IsRegional;
        entity.IsExpress   = value.IsExpress;
        entity.IsIntercity = value.IsIntercity;
        entity.Remark      = value.Remark;

        await trans.CommitTransactionAsync();
    }

    return NoContent();
}

[HttpDelete("{id:int}")]
public async Task<ActionResult> Delete(int id)
{
    using (var trans = _uow.BeginTransaction())
    {
        var entry = await _uow.StationRepository.GetByIdAsync(id);
        if (entry is not null)
        {
            _uow.StationRepository.Remove(entry);
        }
```

```
        await trans.CommitTransactionAsync();

        return NoContent();
    }
}
```

8. EF-Core

8.1. DB-Context

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext() : base() {}
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) :
base(options) {}

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {
        if (!optionsBuilder.IsConfigured) {
            string connectionString =
ConfigurationHelper.GetConfiguration().Get("DefaultConnection", "ConnectionStrings");
            optionsBuilder.UseSqlServer(connectionString);
            }
        optionsBuilder.LogTo(message => Debug.WriteLine(message));
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        modelBuilder.Entity<Child>().Map();
        modelBuilder.Entity<Letter>().Map();
        modelBuilder.Entity<Wish>().Map();
        base.OnModelCreating(modelBuilder);
    }
}
```

8.2. Fluent API

```
public static void Map(this EntityTypeBuilder<Letter> entity)
{
  entity.ToTable("Letter");
  entity.HasKey(l => l.Id);
  entity.Property(l => l.Location).IsRequired().HasMaxLength(255);
  entity.HasOne(l => l.Child).WithMany(c => c.Letters).HasForeignKey(l => l.ChildId);
  entity.HasIndex(l => l.Name).IsUnique();
}
```

8.3. Data Annotations

```
[Timestamp]
public byte[]? RowVersion { get; set; }
```

8.4. Anderes

```
return await Context.Set<Child>().SingleOrDefaultAsync(c => c.Name == childName);
```

```
return await DbSet.Include(e => e.ExamQuestions).FirstOrDefaultAsync(e => e.Name ==
name && e.Date == date);
```

```
return await DbSet.Include(c => c.Races)
            .Where(c => c.Races!.Count > 0)
            .Select(c => new CompetitionSummary()
        {
            Id = c.Id,
            Name = c.Name,
            RaceCount = c.Races!.Count,
            RaceStartTime = c.Races!.Select(r => r.RaceStartTime).Min(),
            RaceEndTime = c.Races!.Select(r => r.RaceStartTime).Max(),
        }).ToListAsync();
```

9. Authentication / Authorization

9.1. Program.cs

```
using System.Text;

using Logic;
using Logic.Abstraction;

using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;

using WebAPI;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
```

```
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "WebAPI", Version = "v1" });

    var xmlFile = "WebAPI.xml";
    var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
    c.IncludeXmlComments(xmlPath);

    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme()
    {
        Name         = "Authorization",
        Type         = SecuritySchemeType.ApiKey,
        Scheme       = "Bearer",
        BearerFormat = "JWT",
        In           = ParameterLocation.Header,
        Description  = "JWT Authorization header using the Bearer scheme. \r\n\r\n" +
                       "Enter 'Bearer' [space] and then your token in the text input
below.\r\n\r\n" +
                       "Example: \"Bearer 1safsfsdfdfd\""
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                {
                    Type = ReferenceType.SecurityScheme,
                    Id   = "Bearer"
                }
            },
            new string[] { }
        }
    });
});

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer           = true,
            ValidateAudience         = true,
            ValidateLifetime         = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer              = builder.Configuration["Jwt:Issuer"],
            ValidAudience            = builder.Configuration["Jwt:Issuer"],
            IssuerSigningKey         = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]!))
        };
    });
```

```
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("Admin", policy => policy.RequireClaim("IsAdmin"));
});

builder.Services.AddTransient<UserManager>();
builder.Services.AddTransient<IMealService,MealService>();
builder.Services.AddTransient<IOrderService, OrderService>();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

// Add CORS to support Single Page Apps (SPAs)
app.UseCors(b => b.AllowAnyHeader().AllowAnyMethod().AllowAnyOrigin());

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

9.2. Login Controller

```
/// <summary>
/// Login with username and password.
/// </summary>
/// <param name="login">Username, password, ...</param>
/// <returns>The JWT with claims.</returns>
[AllowAnonymous]
[HttpPost]
public IActionResult Login([FromBody] User login)
{
    IActionResult response = Unauthorized();
    var           user     = _userManager.AuthenticateUser(login);

    if (user != null)
    {
        var tokenString = _userManager.GenerateJwt(user);
        response = Ok(new { accessToken = tokenString, username = login.Username });
    }

    return response;
```

```
}
```

9.3. JWT & Auth

```
/// <summary>
/// Get the JWT.
/// </summary>
/// <param name="userInfo"></param>
/// <returns></returns>
public string GenerateJwt(UserInfo userInfo)
{
    var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]!));
    var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

    var claims = new List<Claim>()
    {
        new Claim(JwtRegisteredClaimNames.Sub,   userInfo.Username),
        new Claim(JwtRegisteredClaimNames.Email, userInfo.EmailAddress),
        new Claim(JwtRegisteredClaimNames.Jti,   Guid.NewGuid().ToString())
    };

    if (!string.IsNullOrEmpty(userInfo.Roles))
    {
        claims.Add(new Claim(ClaimTypes.Role, userInfo.Roles));
    }

    var token = new JwtSecurityToken(_configuration["Jwt:Issuer"],
        _configuration["Jwt:Issuer"],
        claims,
        expires: DateTime.Now.AddMinutes(120),
        signingCredentials: credentials);

    return new JwtSecurityTokenHandler().WriteToken(token);
}

/// <summary>
/// User authentication.
/// </summary>
/// <param name="login"></param>
/// <returns></returns>
public UserInfo? AuthenticateUser(User login)
{
    var userinfo = _adminUsers.FirstOrDefault(user => user.Username ==
login.Username);
    if (userinfo is not null && login.Password == "McDagobert")
    {
        return userinfo;
```

```
    }

    if (login.Username.Length > 5 && login.Password == "Guest1234")
    {
        return new UserInfo { Username = login.Username, EmailAddress =
"guest@htl.at", Roles = "Guest" };

    }

    return null;
}
```

9.4. Other Controller

```
[Authorize]
[HttpGet]
public async Task<ActionResult<IEnumerable<Meal>>> Get()
{
    return await _mealService.GetMeals();
}
```

2. Angular

1. Befehle

```
# Start application
npx ng serve

# Generate new Component
npx ng g c components/name
# od.
npx ng generate component components/name
```

2. ngOnInit

```
export class ShippingCompanyComponent implements OnInit {
  company: ShippingCompanyDto = { id: 0, name: "" };
  id: number = 0;

  constructor(private cruiserService: CruiserService, private router: Router, private
route: ActivatedRoute) {
  }

  ngOnInit(): void {
```

```
    // route id
    this.route.params.subscribe(params =>
      this.id = +params["id"]
    );

    // get data from service
    this.cruiserService.getCompany(1).subscribe({
      next: data => this.company = data,
      error: data => console.error("Error: ", data)
    });

    // change page title
    this.router.events.subscribe((event) => {
        if (event instanceof NavigationEnd) {
            if (event.url.includes('edit-shipping-company')) {
                this.pageTitle = 'Ship Company';
            } else {
                this.pageTitle = 'This must be changed';
            }
        }
    });
  }
}
```

3. Routing

```
// app-routing.module.ts
const routes: Routes = [
  {path: 'shipping-companies', component: ShippingCompaniesComponent},
  {path: '', redirectTo: '/shipping-companies', pathMatch: 'full'},
  {path: 'shipping-company/:id', component: ShippingCompanyComponent},
];

// Route from page to another when updating
updateCompany() {
    this.cruiserService.updateCompany(this.company).subscribe(() => {
      this.router.navigate(['/shipping-company', this.company.id]);
    });
}

// to use routing you have to use
<router-outlet />
```

4. @for, @if & *ngFor, *ngIf

```
@if (loggedIn) {
  The user is logged in
```

```
} @else {
  The user is not logged in
}

@for (user of users; track user.id) {
  {{ user.name }}
} @empty {
  Empty list of users
}
```

```
<ul>
  <li *ngFor="let user of users; trackBy: trackByUserId">
    {{ user.name }}
  </li>
</ul>
<div *ngIf="users.length === 0">
  Empty list of users
</div>
```

5. Divese Komponenten

5.1. Formulare

```
<form class="mt-3" (ngSubmit)="onSubmit(editDetailForm)" #editDetailForm="ngForm">
  <div class="form-group">
    <label>Name:</label>
    <input class="form-control" id="name" required type="text"
[(ngModel)]="detail.name" name="name" #name="ngModel" />
    <div [hidden]="name.valid || !triedToSubmit" class="alert alert-danger">
      Name is required!
    </div>
  </div>

  <div class="form-group">
    <label>Description:</label>
    <input class="form-control" id="description" required type="text"
[(ngModel)]="detail.description" name="description" #description="ngModel" />
    <div [hidden]="description.valid || !triedToSubmit" class="alert alert-danger">
      Description is required!
    </div>
  </div>

  <div class="form-group">
    <label>Creation Date:</label>
    <input class="form-control" id="creationDate" required type="date"
[(ngModel)]="detail.creationDate" name="creationDate" #creationDate="ngModel" />
    <div [hidden]="creationDate.valid || !triedToSubmit" class="alert alert-danger">
```

```
      Creation Date is required!
    </div>
  </div>

  <div class="form-group">
    <button type="submit" class="btn btn-primary mt-3">Update Detail</button>
  </div>
</form>
```

```
onSubmit(form: NgForm) {
    console.log('Submitting form', form);
    this.triedToSubmit = true;
    if (form.valid) {
        console.log('Form is valid, making API call');
        this.simulationService.updateDetail(this.detail).subscribe(response => {
            console.log('Detail updated successfully', response);
            this.router.navigate(['/simulation', this.id]);
        }, error => {
            console.error('Error updating detail', error);
        });
    }
}
```

5.2. Tabellen

```
<div class="container mt-3">
<h2>Shipping Companies</h2>

<table class="table table-sm">
    <thead>
    <tr>
        <th>Name</th>
        <th>No of Ships</th>
    </tr>
    </thead>
    <tbody>
    <!-- <ng-container *ngFor="let station of stations"> -->
    @for (simulation of simulations; track simulation.id) {
        <tr>
            <td>{{ simulation.name }}</td>
            <td>{{simulation.samples }}</td>
            <td><button class="btn btn-info btn-sm"
(click)="showDetail(simulation)">Details</button></td>
        </tr>
    }
    </tbody>
</table>
```

```
</div>
```

6. Two-Way-Binding

HTML:

```
<app-sizer [(size)]="fontSizePx"></app-sizer>
<div [style.font-size.px]="fontSizePx">Resizable Text</div>
```

TS:

```
fontSizePx = 16;
```

7. Services

7.1. HTTP-Service

Get / Post / Update

```
@Injectable({
  providedIn: 'root',
})
export class CruiserService {
  private baseUrl= "http://localhost:5000";

  constructor(private http: HttpClient) {}

  getShippingCcompanies(): Observable<CompanyOverview[]> {
    return
this.http.get<CompanyOverview[]>(`${this.baseUrl}/ShippingCompany/overview`);
  }

  updateCompany(company: ShippingCompanyDto) {
    const url = `${this.apiUrl}/ShippingCompany/${company.id}`;
    return this.http.put(url, company);
  }
}

// app.module.ts
imports: [
    HttpClientModule,
    // ...
]
```

7.2. auth.guard.ts

```
@Injectable({
  providedIn: 'root'
})
export class AuthGuard {
  constructor(public router: Router) { }

  canActivate(): boolean {
    const jwt = sessionStorage.getItem('jwt');
    if (!jwt) {
      this.router.navigate(['login']);
      return false;
    }
    return true;
  }
}
```

7.3. Data Service

```
@Injectable({
  providedIn: 'root'
})
export class DataService {

  loggedInUser: string | undefined;
  baseUrl = 'http://localhost:5124/api';

  constructor(private httpClient: HttpClient) {
    this.baseUrl = environment.apiBaseUrl;
    const jwt = sessionStorage.getItem('jwt');
    if (jwt) {
      const user = sessionStorage.getItem('McDagobert.Login-user');
      this.loggedInUser = user ? JSON.parse(user) : undefined;
    }
  }

  login(usernameInput: string, passwordInput: string): Observable<AuthResponse> {
    const url = `${this.baseUrl}/login`;
    const user: User = {
      username: usernameInput,
      password: passwordInput
    };
    return this.httpClient.post<AuthResponse>(url, user);
  }
  getMeals(): Observable<Meal[]> {
    const url = `${this.baseUrl}/Meal`;
    return this.httpClient.get<Meal[]>(url);
  }
```

```
}
```

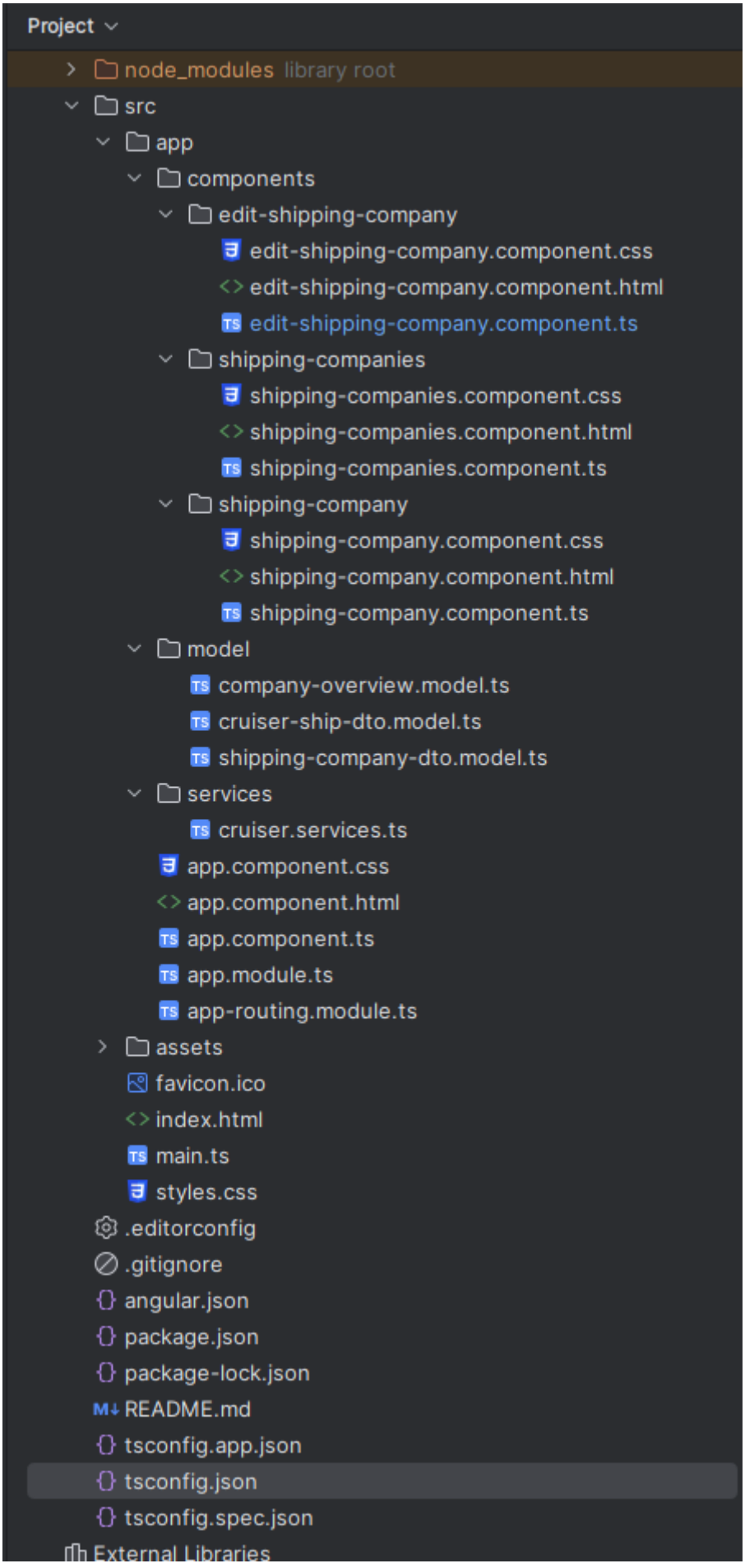
7.4. Auth Inspector

```
import { environment } from '../../environments/environment';
import { HttpInterceptorFn } from '@angular/common/http';

export const authInterceptor: HttpInterceptorFn = (request, next) => {
  console.log('intercepted request', request);
  const jwt = sessionStorage.getItem('jwt');
  const isLoggedIn = jwt;
  const isApiUrl = request.url.startsWith(environment.apiBaseUrl);
  console.log('intercepted request Logged in', isLoggedIn, jwt, isApiUrl);
  if (isLoggedIn && isApiUrl) {
    console.log('intercepted request Logged in');
    request = request.clone({
      setHeaders: { Authorization: `Bearer ${jwt}` }
    });
  }

  return next(request);
}
```

8. Input & Output

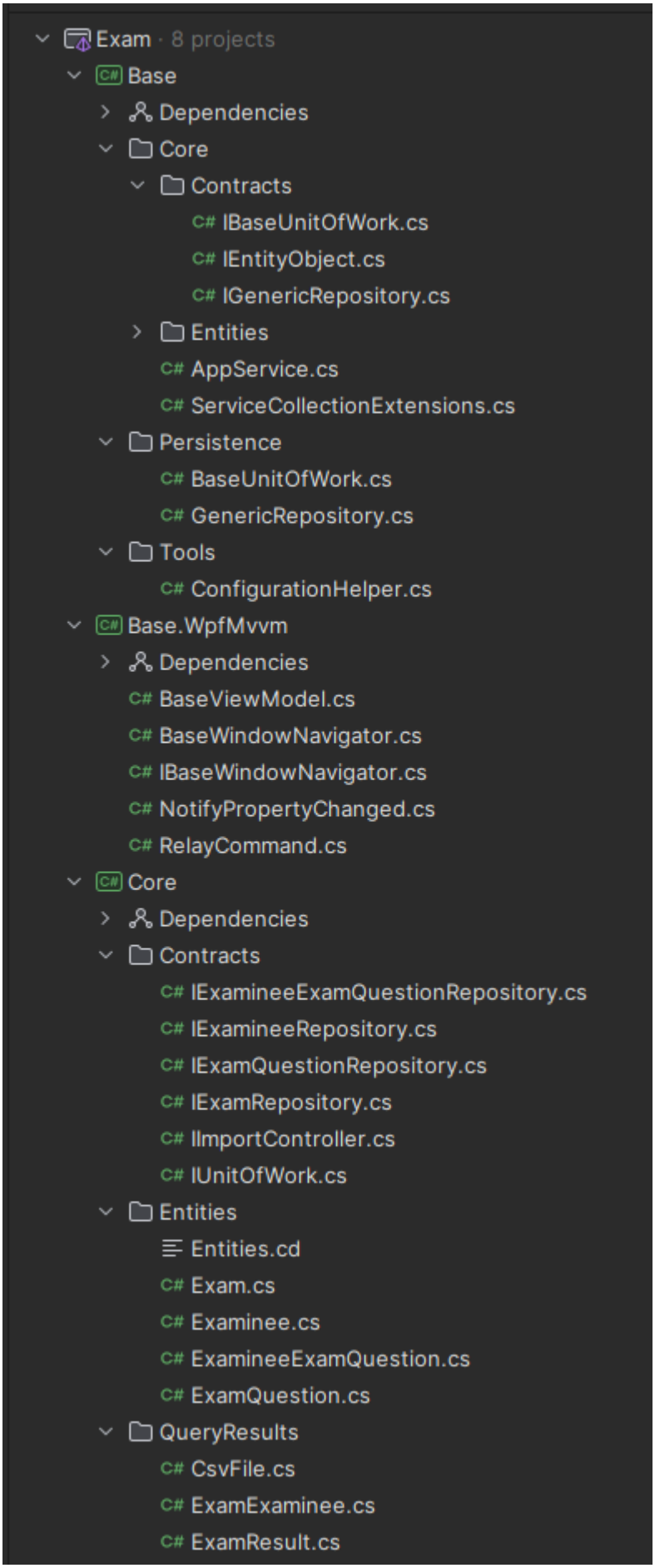
```
// Input Usage Child
@Input() mealOrder!: MealOrder;

// Input Usage Parent:
<app-meal-order [mealOrder]="order"></app-meal-order>

// Output Usage Child
@Output() mealOrderChange = new EventEmitter<MealOrder>();
this.mealOrderChange.emit(order);

// Output Usage Parent
<app-meal-order (mealOrderChange)="onOrderChange($event)"></app-meal-order>
```

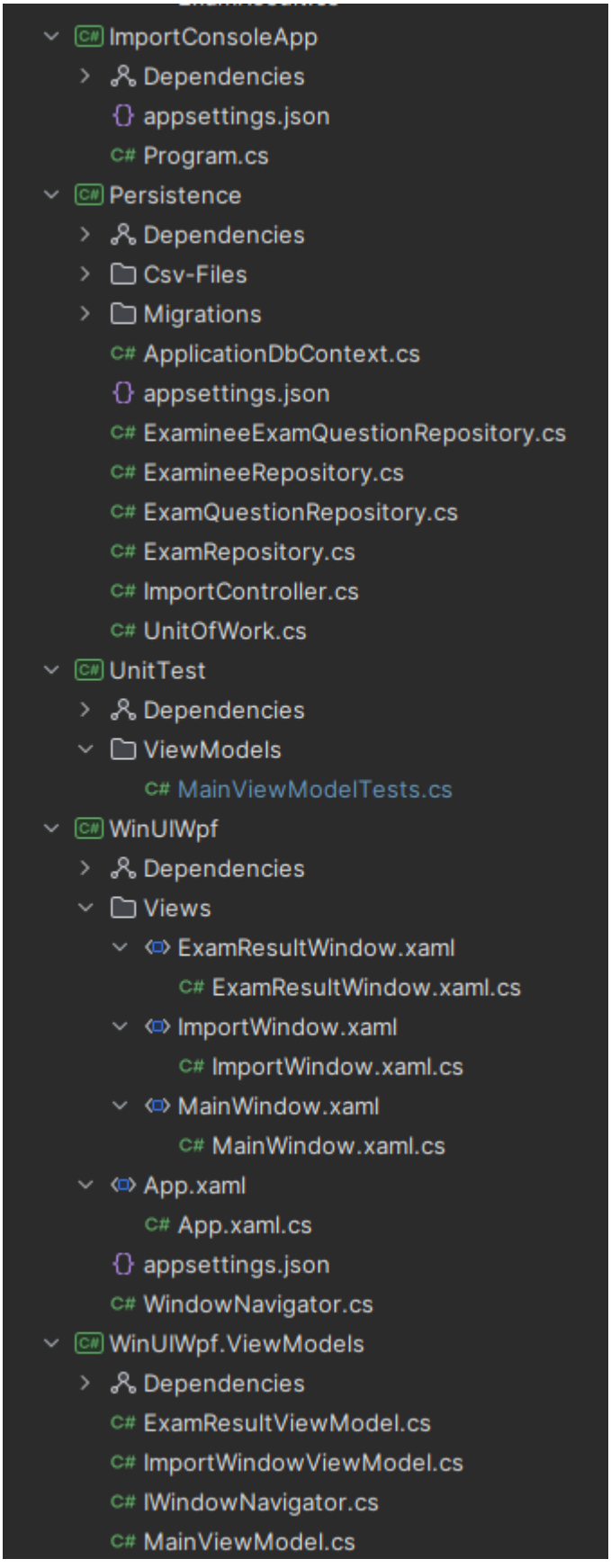
9. File Structure


Project
node_modules library root
src
app
components
edit-shipping-company
edit-shipping-company.component.css
edit-shipping-company.component.html
edit-shipping-company.component.ts
shipping-companies
shipping-companies.component.css
shipping-companies.component.html
shipping-companies.component.ts
shipping-company
shipping-company.component.css
shipping-company.component.html
shipping-company.component.ts
model
company-overview.model.ts
cruiser-ship-dto.model.ts
shipping-company-dto.model.ts
services
cruiser.services.ts
app.component.css
app.component.html
app.component.ts
app.module.ts
app-routing.module.ts
assets
favicon.ico
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
package.json
package-lock.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
External Libraries

3. WPF

1. Struktur


Exam · 8 projects
Base
Dependencies
Core
Contracts
IBaseUnitOfWork.cs
IEntityObject.cs
IGenericRepository.cs
Entities
AppService.cs
ServiceCollectionExtensions.cs
Persistence
BaseUnitOfWork.cs
GenericRepository.cs
Tools
ConfigurationHelper.cs
Base.WpfMvvm
Dependencies
BaseViewModel.cs
BaseWindowNavigator.cs
IBaseWindowNavigator.cs
NotifyPropertyChanged.cs
RelayCommand.cs
Core
Dependencies
Contracts
IExamineeExamQuestionRepository.cs
IExamineeRepository.cs
IExamQuestionRepository.cs
IExamRepository.cs
IImportController.cs
IUnitOfWork.cs
Entities
Entities.cd
Exam.cs
Examinee.cs
ExamineeExamQuestion.cs
ExamQuestion.cs
QueryResults
CsvFile.cs
ExamExaminee.cs
ExamResult.cs



1.2. Lists in Model vs Logic

```
// Model
public  ObservableCollection<Word> Words { get; set; } = new();
```

```
// Logic
public IEnumerable<Word> Words { get; set; }
```

2. MainViewModel

```
public class MainViewModel : BaseViewModel
{
    #region ctr

    public MainViewModel(IUnitOfWork uow)
    {
        _uow = uow;
    }

    private IUnitOfWork _uow;

    #endregion

    #region Properties

    public IWindowNavigator? Controller { get; set; }

    public ObservableCollection<Exam> Exams        { get; set; } = new
ObservableCollection<Exam>();
    public Exam?                      SelectedExam { get; set; }

    public RelayCommand ShowExamResultCommand  => new RelayCommand(async _ => await
ShowExamResult(), (_) => SelectedExam != null);
    public RelayCommand ImportWindowCommand => new RelayCommand(async _ => await
OpenImportWindow());

    #endregion

    #region Operations

    public async Task LoadDataAsync()
    {
        Exams.Clear();
        var exams = await _uow.Exam.GetNoTrackingAsync();
        foreach (var v in exams)
        {
            Exams.Add(v);
        }
    }

    private async Task ShowExamResult()
    {
        await Task.CompletedTask;
        Controller?.ShowExamResultWindow(SelectedExam!.Id);
```

```
    }

    private async Task OpenImportWindow()
    {
        Controller?.ShowImportWindow();
        await LoadDataAsync();
    }

    #endregion
}
```

2.1 Message Box

```
var result =
    MessageBox.Show("Update booking. Check in date: {SelectedBooking.Start} -->
{_start}, Check out date: {SelectedBooking.End} --> {_end} ",
    "Question",
    MessageBoxButton.YesNo, MessageBoxImage.Warning);

        if (result == MessageBoxResult.Yes)
        {
            //...
        }
```

2.2 Filter

```
public async Task Filter()
{
   var rooms = UnfilteredRoomSummaries
        .Where(r =>
            (RoomTypeFilter.Id == -1 || r.Type == RoomTypeFilter.Type)
        && (RoomSpecialFilter.Id == -1 ||
r.Specials.Contains(RoomSpecialFilter.Special) ))
        .ToList();
   Rooms.Clear();
   // Do NOT use foreach
   foreach (var roomSummary in rooms)
   {
       Rooms.Add(roomSummary);
   }
}
```

2.3 Update

Repository:

```
public async Task UpdateAsync(HikeDto hikeDto)
{
    var hike = await DbSet.Include(h => h.Companions).Include(h =>
h.Highlights).FirstOrDefaultAsync( h => h.Id == hikeDto.Id);

    if (hike == null)
    {
        throw new ArgumentException("Hike not found");
    }

    hike.Distance = hikeDto.Distance;
    hike.Duration = hikeDto.Duration;
    hike.Trail = hikeDto.Trail;

    DbSet.Update(hike);
}
```

Or ViewModel:

```
var inDb = (await _uow.RailwayCompanyRepository.GetByIdAsync(SelectedCompany!.Id)) ??
throw new ArgumentNullException();

inDb.City     = City;
inDb.PLZ      = PLZ;
inDb.Street   = Street;
inDb.StreetNo = StreetNo;
await _uow.SaveChangesAsync();

await InitializeDataAsync();
IsEditMode = !IsEditMode;
```

2.3 Load

```
public async Task LoadDataAsync()
{
    var r = await uow.Room.GetRoomSummaryListAsync();
    Rooms.Clear();
    foreach (var roomSummary in r)
    {
        Rooms.Add(roomSummary);
    }

    UnfilteredRoomSummaries = r;

    var types = await uow.RoomType.GetAsync();
    RoomTypes.Clear();
```

```
    var defaultFilterRoomType = new RoomType()
    {
        Type = "<Alle>",
        Id = -1
    };
    RoomTypes.Add(defaultFilterRoomType);
    RoomTypeFilter = defaultFilterRoomType;
    foreach (var roomType in types)
    {
        RoomTypes.Add(roomType);
    }

    var specials = await uow.RoomSpecial.GetAsync();
    RoomSpecials.Clear();

    var defaultFilterSpecial = new RoomSpecial()
    {
        Id = -1,
        Special = "<Alle>"
    };

    RoomSpecials.Add(defaultFilterSpecial);
    RoomSpecialFilter = defaultFilterSpecial;

    foreach (var roomSpecial in specials)
    {
        RoomSpecials.Add(roomSpecial);
    }
}
```

2.3.1. Load with Filter / NoTrackingAsync

```
public async Task Load(IUnitOfWork uow)
{
    Expression<Func<Station, bool>>? filter = null;

    if (SelectedCity is not null)
    {
        filter = x => x.CityId == SelectedCity.Id;
    }

    FilteredStations.Clear();
    (await _uow.StationRepository.GetNoTrackingAsync(filter:
filter)).ToList().ForEach(s => FilteredStations.Add(s));
}
```

2.3.2. Load by Id

```
var station = await uow.StationRepository.GetByIdAsync(StationId!.Value,
nameof(Station.RailwayCompanies), nameof(Station.Infrastructures),
nameof(Station.Lines), nameof(Core.Entities.City));
```

3. Window Navigator

Anzeigen von Fenster:

```
public void ShowExamResultWindow(Exam Exam)
{
    using (var scope = AppService.ServiceProvider.CreateScope())
    {
        var showExamResultsWindow    =
scope.ServiceProvider.GetRequiredService<ExamResultWindow>();
        var viewModel = (ResultWindowViewModel)showExamResultsWindow.DataContext;
        viewModel.Exam = Exam;
        showExamResultsWindow.ShowDialog();
    }
}
```

IWindowNavigator.cs:

```
public interface IWindowNavigator : IBaseWindowNavigator
{
    public void ShowImportWindow();
    public void ShowExamResultWindow(int examId);
}
```

WindowNavigator.cs:

```
public class WindowNavigator : BaseWindowNavigator, IWindowNavigator
{
    public WindowNavigator(Window window) : base(window)
    {
    }

    public void ShowImportWindow() { /*...*/ }

    public void ShowExamResultWindow(int examId)
    {
        using (var scope = AppService.ServiceProvider!.CreateScope())
        {
            var examResultWindow =
scope.ServiceProvider.GetRequiredService<ExamResultWindow>();
```

```
            var examResultVm     = examResultWindow.DataContext as
ExamResultViewModel;
            examResultVm.ExamId = examId;
            examResultWindow.ShowDialog();
        }
    }
}
```

Close Window:

```
Controller!.Close();
```

4. ImportConsole

```
AppService.ServiceCollection = new ServiceCollection();
AppService.ServiceCollection
    .AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(connectionString))
    .AddScoped<IUnitOfWork, UnitOfWork>()
    .AddTransient<IDriverRepository, DriverRepository>()
    .AddScoped<IImportService, ImportService>();
```

```
 using (var scope = AppService.ServiceProvider!.CreateScope())
 {
    var importService = scope.ServiceProvider.GetRequiredService<IImportService>();
    var uow = scope.ServiceProvider.GetRequiredService<IUnitOfWork>();

    foreach (CsvContent race in await new
CsvImport<CsvContent>().ReadAsync("ImportData/Races.csv"))
    {
        await import.ImportRace(race);
        await uow.SaveChangesAsync();
    }
}
```

4.1 ImportService

```
public class ImportService : IImportService
{
    private readonly IUnitOfWork _uow;

    public ImportService(IUnitOfWork uow)
    {
        _uow = uow;
    }
```

```
    public async Task ImportRace(CsvContent csvContent)
    {
        await uow.Race.AddAsync(r);
    }
```

5. XAML

5.1. Allgemein

App.xaml

```
<Application x:Class="WinUIWpf.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:WinUIWpf"
             Startup="AppStartup">
    <Application.Resources>
        <Style TargetType="TextBlock" x:Key="Header">
            <Setter Property="FontSize" Value="30" />
            <Setter Property="Padding" Value="10,5,10,10" />
            <Setter Property="VerticalAlignment" Value="Center"></Setter>
        </Style>
        <Style TargetType="TextBlock" x:Key="HeaderSubtitle" BasedOn="{StaticResource
Header}">
            <Setter Property="FontSize" Value="15" />
            <Setter Property="HorizontalAlignment" Value="Left" />
        </Style>
        <Style TargetType="Button" x:Key="ActionButton">
            <Setter Property="FontSize" Value="14"></Setter>
            <Setter Property="Height" Value="30"></Setter>
            <Setter Property="Margin" Value="10,5,10,10" />
            <Setter Property="Padding" Value="5"></Setter>
        </Style>
        <Style TargetType="Button" x:Key="DeleteButton" BasedOn="{StaticResource
ActionButton}">
            <Setter Property="Background" Value="Red"></Setter>
        </Style>
    </Application.Resources>
</Application>
```

App.xaml.cs (Dependency Injection)

```
public partial class App : Application
{
    private void AppStartup(object sender, StartupEventArgs e)
    {
        var configuration = ConfigurationHelper.GetConfiguration();
```

```
        var connectionString = configuration.GetConnectionString("DefaultConnection")
?? throw new InvalidOperationException("Connection string 'DefaultConnection' not
found.");

        AppService.ServiceCollection = new ServiceCollection();
        AppService.ServiceCollection
            .AddSingleton(configuration)
            .AddAssemblyByName(
                n => n.EndsWith("ViewModel"),
                ServiceLifetime.Transient,
                typeof(ViewModels.MainViewModel).Assembly)
            .AddDbContext<ApplicationDbContext>(options =>
                options.UseSqlServer(connectionString))
            .AddScoped<IUnitOfWork, UnitOfWork>()
            .AddTransient<IExamineeRepository, ExamineeRepository>()
            .AddTransient<IImportController, ImportController>()
            .AddTransient<MainWindow>()
            ;

        AppService.BuildServiceProvider();

        MainWindow = AppService.GetRequiredService<MainWindow>();
        MainWindow.Show();
    }
}
```

MainWindow.xaml

```
<Window x:Class="WinUIWpf.Views.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        d:DataContext="{d:DesignInstance Type=viewModels:MainViewModel}"
        xmlns:local="clr-namespace:WinUIWpf"
        xmlns:viewModels="clr-
namespace:WinUIWpf.ViewModels;assembly=WinUIWpf.ViewModels"
        mc:Ignorable="d"
        Title="Exam" Height="400" Width="400">

    <Window.Resources>
        <Style x:Key="GridColumn" TargetType="TextBlock">
            <Setter Property="FontSize" Value="12"></Setter>
            <Setter Property="Padding" Value="5"></Setter>
        </Style>
        <Style x:Key="NumberColumn" TargetType="TextBlock" BasedOn="{StaticResource
GridColumn}">
            <Setter Property="HorizontalAlignment" Value="Center"></Setter>
        </Style>
```

```
        <Style x:Key="TextColumn" TargetType="TextBlock" BasedOn="{StaticResource
GridColumn}">
            <Setter Property="HorizontalAlignment" Value="Left"></Setter>
            <Setter Property="VerticalAlignment" Value="Center"></Setter>
        </Style>
        <Style x:Key="CheckboxColumn" TargetType="CheckBox">
            <Setter Property="Padding" Value="5"></Setter>
            <Setter Property="HorizontalAlignment" Value="Center"></Setter>
            <Setter Property="VerticalAlignment" Value="Center"></Setter>
        </Style>
    </Window.Resources>
    <DockPanel>
    </DockPanel>
</Window>
```

MainWindow.xaml.cs

```
public partial class MainWindow : Window
{
    public MainWindow(MainViewModel viewModel)
    {
        InitializeComponent();
        viewModel.Controller = new WindowNavigator(this);
        DataContext          = viewModel;

        Loaded += async (v, e) =>
        {
            await (DataContext as MainViewModel)!
                .LoadDataAsync();
        };
    }
}
```

GameWindow.xaml

```
<Window x:Class="WPFApp.Views.GameWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:WPFApp.Views"
        xmlns:viewModel="clr-namespace:WPFApp.ViewModel"
        mc:Ignorable="d"
        Title="GameWindow" Height="350" Width="1000">
    <Window.DataContext>
        <viewModel:GameViewModel/>
    </Window.DataContext>
</Window>
```

Wichtig: DataContext setzen!

GameWindow.xaml.cs

```
public partial class GameWindow
{
    public GameWindow()
    {
        InitializeComponent();
        DataContext = new GameViewModel(); // maybe not needed if already in xml
    }
}
```

5.2. Buttons & Binding

```
<TextBox Text="{Binding NewWord, UpdateSourceTrigger=PropertyChanged,
StringFormat='{}{0:hh:mm:ss dd.MM.yyyy}'}" MinWidth="300" />
<Button Content="Apply" Command="{Binding SaveCommand}" MinWidth="100" />
```

```
private string _newWord = String.Empty;
public string NewWord
{
  get => _newWord ;
  set { _newWord = value; OnPropertyChanged(); }
}

public GameViewModel()
{
    SaveCommand = new DelegateCommand(Save, CanSave);
}

public ICommand SaveCommand {get;}
```

Vertical Button Group:

```
<DockPanel>
    <StackPanel HorizontalAlignment="Left" VerticalAlignment="Top"
DockPanel.Dock="Top">
        <StackPanel Orientation="Horizontal">
            <Button Content="ScaleX +" Width="80" Command="{Binding IncreaseScaleX}"
/>
            <Button Content="ScaleX -" Width="80" Command="{Binding DecreaseScaleX}"
/>
            <Button Content="ScaleY +" Width="80" Command="{Binding IncreaseScaleY}"
/>
            <Button Content="ScaleY -" Width="80" Command="{Binding DecreaseScaleY}"
```

```
/>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Button Content="OfsX +" Width="80" Command="{Binding IncreaseOffsetX}" />
            <Button Content="OfsX -" Width="80" Command="{Binding DecreaseOffsetX}" />
            <Button Content="OfsY +" Width="80" Command="{Binding IncreaseOffsetY}" />
            <Button Content="OfsY -" Width="80" Command="{Binding DecreaseOffsetY}" />
        </StackPanel>
    </StackPanel>
</DockPanel>
```

Button with Styling:

```
<Window.Resources>
    <Style x:Key="Button" TargetType="Button">
        <Setter Property="HorizontalAlignment" Value="Left"></Setter>
        <Setter Property="VerticalAlignment" Value="Top"></Setter>
        <Setter Property="Width" Value="100"></Setter>
        <Setter Property="Height" Value="20"></Setter>
        <Setter Property="Margin" Value="5"></Setter>
    </Style>
</Window.Resources>

<Button Grid.Row="2" Grid.ColumnSpan="2" Style="{StaticResource Button}"
Command="{Binding ImportCommand}"> Import Csv</Button>
```

RelayCommand:

```
namespace Sudoku.Tools;

using System;
using System.Windows.Input;

public class RelayCommand : ICommand
{
    private readonly Action<object?>     _execute;
    private readonly Predicate<object?>? _canExecute;

    public RelayCommand(Action<object?> execute, Predicate<object?>? canExecute =
null)
    {
        _execute    = execute;
        _canExecute = canExecute;
    }

    public bool CanExecute(object? parameter)
    {
        return _canExecute == null || _canExecute(parameter);
    }
```

```
    public event EventHandler? CanExecuteChanged
    {
        add => CommandManager.RequerySuggested += value;
        remove => CommandManager.RequerySuggested -= value;
    }

    public void Execute(object? parameter) => _execute(parameter);
}
```

```
public ObservableCollection<CsvFile> CsvFiles { get; set; } = new
ObservableCollection<CsvFile>();

public CsvFile? Selected { get; set; }

public RelayCommand ImportCommand    => new RelayCommand(async _ => await
ImportFileAsync(), (_) => Selected != null);
```

5.3. DataGrid

```
<DataGrid
    Margin="20 0"
    MinHeight="210"
    ItemsSource="{Binding CsvFiles}"
    AutoGenerateColumns="False"
    ScrollViewer.VerticalScrollBarVisibility="Visible"
    ScrollViewer.HorizontalScrollBarVisibility="Auto"
    SelectedItem="{Binding SelectedCsv}"
    MaxHeight="600">
    <DataGrid.Columns>
            <DataGridTextColumn Binding="{Binding ExamName}" Header="Name"
MinWidth="50"
                                IsReadOnly="True" />
            <DataGridTextColumn Binding="{Binding Date,
StringFormat={}{0:yyyy.MM.dd}}" Header="Datum" MinWidth="100"
                                IsReadOnly="True" />
            <DataGridTextColumn Binding="{Binding ExamineeName}" Header="Examenee"
MinWidth="50"
                                IsReadOnly="True" />
            <DataGridTextColumn Binding="{Binding FileName}" Header="File"
MinWidth="250"
                                IsReadOnly="True" />
    </DataGrid.Columns>
</DataGrid>
```

5.4. Stack Panel with Content & Separator

```
<Separator></Separator>
<StackPanel Orientation="Horizontal" DockPanel.Dock="Bottom">
    <Label Content="New word" MinWidth="100" />
    <TextBox Text="{Binding Sentence.NewWord, UpdateSourceTrigger=PropertyChanged}"
MinWidth="300" />
    <Button Content="Apply" Command="{Binding SaveCommand}" MinWidth="100" />
</StackPanel>
```

5.5. UserControl

```
<UserControl x:Class="SinusUserControl.SinusControl"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300" />
```

5.6. DockPanel

```
----ok
<DockPanel>
    <Button DockPanel.Dock="Top" Content="Top" />
    <Button DockPanel.Dock="Bottom" Content="Bottom" />
    <Button DockPanel.Dock="Left" Content="Left" />
    <Button DockPanel.Dock="Right" Content="Right" />
    <Button Content="Center" />
</DockPanel>
----
```

UniformGrid:

```
<UniformGrid Grid.Row="3" Columns="4">
    <Button Content="Close" Margin="5" Command="{Binding CloseCommand}" />
    <Button Content="Edit" Margin="5" Command="{Binding EditCommand}" />
    <Button Content="Update" Margin="5" Command="{Binding UpdateCommand}" />
</UniformGrid>
```

DockPanel with UniformGrid:

```
<DockPanel>
        <UniformGrid DockPanel.Dock ="Top" Columns="1" HorizontalAlignment="Stretch"
Margin="10">
```

```
            <StackPanel Orientation="Horizontal" HorizontalAlignment="Left">
                <TextBlock Style="{StaticResource Header}">Competitions</TextBlock>
            </StackPanel>
        </UniformGrid>

        <UniformGrid DockPanel.Dock="Right" Rows="4">
            <Button Command="{Binding TODOCommand}" Style="{StaticResource
ActionButton}">Races</Button>
        </UniformGrid>

        <DataGrid Margin="10" Grid.Row="2"
                  ItemsSource="{Binding TODO}"
                  SelectedItem="{Binding TODO}"
                  AutoGenerateColumns="False"
                  CanUserAddRows="False"
                  IsReadOnly="True">
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding Path=Name}" Header="Name"
                                    ElementStyle="{StaticResource StringColumn}">
                </DataGridTextColumn>
                <DataGridTextColumn Binding="{Binding Path=RaceCount}" Header="Races"
                                    ElementStyle="{StaticResource NumberColumn}">
                </DataGridTextColumn>
                <DataGridTextColumn Binding="{Binding Path=FirstRace,
StringFormat={}{0:yyyy.MM.dd HH.mm}}" Header="Start"
                                    ElementStyle="{StaticResource DateColumn}">
                </DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>
    </DockPanel>
```

5.7. Slider

```
<Slider Grid.Row="0" Grid.Column="1" Value="{Binding Path=ScaleX}" Minimum="0.1"
                        Maximum="300" Orientation="Horizontal" />
```

5.8. Grid

```
<Grid>
  <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="Auto" />
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
```

```
  </Grid.ColumnDefinitions>

  <Label Grid.Row="0" Grid.Column="0" Content="ScaleX" />
  <Label Grid.Row="1" Grid.Column="0" Content="ScaleY" />

  <Slider Grid.Row="0" Grid.Column="1" Value="{Binding Path=ScaleX}" Minimum="0.1"
          Maximum="300" Orientation="Horizontal" />
  <Slider Grid.Row="1" Grid.Column="1" Value="{Binding Path=ScaleY}" Minimum="0.1"
          Maximum="300" Orientation="Horizontal" />
</Grid>
```

5.9. Implement other XAML into one with variables

MainWindow.xaml

```
<local:SinusControl
            x:Name="_sinus"
            ScaleX="{Binding ScaleX}" />
```

SinusControl.xaml.cs

```
public static DependencyProperty ScaleXProperty =
DependencyProperty.Register(nameof(ScaleX), typeof(double), typeof(SinusControl), new
PropertyMetadata(10.0, OnScaleXChanged));

public double ScaleX
{
    get => (double)GetValue(ScaleXProperty);
    set => SetValue(ScaleXProperty, value);
}

private static void OnScaleXChanged(DependencyObject dependencyObject,
DependencyPropertyChangedEventArgs e)
{
    var ctrl = (SinusControl)dependencyObject;
    ctrl.InvalidateVisual();
}

public SinusControl()
{
    InitializeComponent();
}
```

5.10. ComboBox

```
<ComboBox ItemsSource="{Binding RoomTypes}"
          SelectedItem="{Binding RoomTypeFilter}"
```

```
          DisplayMemberPath="Type"
          SelectedIndex="0"></ComboBox>
<Label>Room Special:</Label>
```

5.11. ListView

```
<ListView Grid.Row="1" Margin="10"
            ItemsSource="{Binding FilteredStations}" SelectedItem="{Binding
SelectedStation}">
    <ListView.View>
        <GridView>
            <GridViewColumn Header="Code" DisplayMemberBinding="{Binding Code}"
Width="70" />
            <GridViewColumn Header="Name" DisplayMemberBinding="{Binding Name}"
Width="100" />
            <GridViewColumn Header="City" DisplayMemberBinding="{Binding City.Name}"
Width="70" />
        </GridView>
    </ListView.View>
</ListView>
```

5.12. Grid with Filter and List

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>

    <!-- Filterbereich -->
    <StackPanel Margin="10" Grid.Row="0" Orientation="Horizontal">
        <Label Content="City:" />
        <!-- ItemSource and DisplayMemberPath="Name" and SelectedValue="{Binding
SelectedCity}" -->
        <ComboBox Width="150" Margin="5" ItemsSource="{Binding Cities}"
DisplayMemberPath="Name" SelectedValue="{Binding SelectedCity}" />

        <Button Content="Filter" Width="60" Margin="10,0,0,0" Command="{Binding
FilterCommand}" />
    </StackPanel>

    <ListView Grid.Row="1" Margin="10"
                ItemsSource="{Binding FilteredStations}" SelectedItem="{Binding
SelectedStation}">
        <ListView.View>
            <GridView>
                <GridViewColumn Header="Code" DisplayMemberBinding="{Binding Code}"
```

```
Width="70" />
                <GridViewColumn Header="Name" DisplayMemberBinding="{Binding Name}"
Width="100" />
                <GridViewColumn Header="City" DisplayMemberBinding="{Binding
City.Name}" Width="70" />
            </GridView>
        </ListView.View>
    </ListView>

    <UniformGrid Grid.Row="2" Columns="4">
        <Button Content="Details" Margin="10" Command="{Binding DetailCommand}" />
    </UniformGrid>
</Grid>
```

6. Drawing

6.1. Define Pens & call OnRender

```
private static readonly Pen RedPen  = new(new SolidColorBrush(Colors.Red),  1.0d);
private static readonly Pen GridPen = new(new SolidColorBrush(Colors.Blue), 0.5d);

protected override void OnRender(DrawingContext drawingContext)
{
    DrawSinus(drawingContext);
}
```

6.2. Draw Line

```
context.DrawLine(GridPen, ToPoint(0, 0), ToPoint(Math.PI*2, 0));

// or

context.DrawLine(BluePen, new Point(x, y), new Point(x2, y2));
```

6.3. Calculate Points for Sinus

```
private double ToX(double x) { return (x + OffsetX) * ScaleX ; }
private double ToY(double y) { return (ActualHeight / 2) - (y+ OffsetY) * ScaleY ; }

private void ToPoint(double x, double y, ref Point pt)
{
    pt.X = ToX(x);
    pt.Y = ToY(y);
}

private Point ToPoint(double x, double y)
```

```
{
    var pt = new Point
    {
        X = ToX(x),
        Y = ToY(y)
    };
    return pt;
}
```

6.4. Get points to Control & Call controll

```
public SimulationDetailWindow(SimulationDetailViewWindowViewModel viewModel)
{
    InitializeComponent();

    viewModel.Controller = new WindowNavigator(this);
    DataContext          = viewModel;

    Loaded += async (v, e) =>
    {
        await (DataContext as SimulationDetailViewWindowViewModel)!
            .InitializeDataAsync();

        SimulationPreviewControl.Samples = viewModel.Samples;
    };
}
```

```
<controls:SimulationPreviewControl x:Name="SimulationPreviewControl" Grid.Row="3"
Margin="0 20" />
```

6.4. Draw spots to specific points

```
public partial class SimulationPreviewControl : UserControl
{
    public SimulationPreviewControl() {InitializeComponent();}

    public static DependencyProperty SamplesProperty =
DependencyProperty.Register(nameof(Samples), typeof(List<(double X, double Y, double
Value)>), typeof(SimulationPreviewControl),
        new PropertyMetadata(new List<(double X, double Y, double Value)>(),
OnSamplesChanged));

    public IList<(double X, double Y, double Value)> Samples
    {
        get => (IList<(double X, double Y, double Value)>)GetValue(SamplesProperty);
        set => SetValue(SamplesProperty, value);
    }
```

```
    private static void OnSamplesChanged(DependencyObject dependencyObject,
DependencyPropertyChangedEventArgs e)
    {
        var ctrl = (SimulationPreviewControl)dependencyObject;
        ctrl.InvalidateVisual();
    }

    private static readonly Pen RedPen = new Pen(new SolidColorBrush(Colors.Red),
1.0d);

    protected override void OnRender(DrawingContext drawingContext)
    {
        DrawSamples(drawingContext);
    }

    private double _scaleX  = 1.0;
    private double _scaleY  = 1.0;
    private double _offsetX = 1.0;
    private double _offsetY = 1.0;

    double ToX(double x){ return (x + _offsetX) * _scaleX;}

    double ToY(double y) { return ActualHeight - ((y + _offsetY) * _scaleY); }

    private void DrawSamples(DrawingContext context)
    {
        if (Samples == null || Samples.Count == 0)
            return;

        var minX = Samples.Min(pt => pt.X);
        var minY = Samples.Min(pt => pt.Y);

        var maxX = Samples.Max(pt => pt.X);
        var maxY = Samples.Max(pt => pt.Y);

        var sizeX = maxX - minX;
        var sizeY = maxY - minY;

        _scaleX = ActualWidth / sizeX;
        _scaleY = ActualHeight / sizeY;

        _scaleX = Math.Min(_scaleX, _scaleY);
        _scaleY = _scaleX;

        _offsetX = -minX;
        _offsetY = -minY;

        if (_scaleX == 0.0 || _scaleY == 0.0) return;

        foreach (var sample in Samples)
```

```
        {
            double x = ToX(sample.X);
            double y = ToY(sample.Y);
            double pointSize = 5;
            context.DrawEllipse(null, RedPen, new Point(x, y), pointSize, pointSize);
        }
    }
}
```

7. Unit Tests (ViewModel-Tests)

Load Test

```
[Fact]
public void LoadTest()
{
    var lines = new string[2];
    lines[0] = "Test";
    lines[1] = $"test;{DateTime.Now}";

    File.WriteAllLines("test-game.txt", lines);

    var vm = new GameViewModel
    {
        FileName = "test-game.txt"
    };
    vm.Load();

    vm.Sentence.NameOfGame.Should().Be(lines[0]);
}
```

Save Test

```
[Fact]
public void SaveTest()
{
    var vm = new GameViewModel();
    vm.FileName            = "test-game.txt";
    vm.Sentence.NameOfGame = "Test";
    vm.Sentence.Words.Add(new Word()
    {
        Name = "test",
        From = DateTime.Now
    });

    vm.Save();

    var lines = File.ReadAllLines("test-game.txt");
```

```
    lines[0].Should().Be("Test");
    lines[1].Should().Contain("test");
}
```

Button Inactive Test

```
[Fact]
public async Task ResultButtonInactiveTest()
{
    // Arrange
    var uow = Substitute.For<IUnitOfWork>();
    var mv = new MainViewModel(uow);
    //Act
    mv.SelectedExam = null;
    mv.ShowExamResultCommand.CanExecute(null).Should().BeFalse();
    mv.SelectedExam = new Exam();
    mv.ShowExamResultCommand.CanExecute(null).Should().BeTrue();
}
```

Load Data Test

```
[Fact]
public async Task LoadDataTest()
{
    //Arrange
    var uow     = Substitute.For<IUnitOfWork>();
    var examRep = Substitute.For<IExamRepository>();
    uow.Exam.Returns(examRep);
    var examMaxi = new Exam() { Id = 1, Name = "Maxi"};
    var examSeppi = new Exam() { Id = 2, Name = "Seppi"};

    uow.Exam.GetNoTrackingAsync(Arg.Any<Expression<Func<Exam, bool>>>(),
            Arg.Any<Func<IQueryable<Exam>, IOrderedQueryable<Exam>>>(),
            Arg.Any<string[]>())
        .Returns(new Exam[] { examMaxi, examSeppi });

    var mv = new MainViewModel(uow);

    //Act
    await mv.LoadDataAsync();

    //Assert
    mv.Exams.Should().Contain(examMaxi);
    mv.Exams.Should().Contain(examSeppi);
    mv.Exams.Should().HaveCount(2);
}
```

8. Other

8.1. DelegateCommand

```
namespace WPFApp.Helpers
{
    public class DelegateCommand : ICommand
    {
        private readonly Action     _command;
        private readonly Func<bool> _canExecute;

        public event EventHandler CanExecuteChanged
        {
            add => CommandManager.RequerySuggested    += value;
            remove => CommandManager.RequerySuggested -= value;
        }

        public DelegateCommand(Action command, Func<bool> canExecute = null)
        {
            if (command == null)
                throw new ArgumentNullException();
            _canExecute = canExecute;
            _command    = command;
        }

        public void Execute(object parameter)
        {
            _command();
        }

        public bool CanExecute(object parameter)
        {
            return _canExecute == null || _canExecute();
        }
    }
}
```

8.2. BindableBase & Model

WpfApp/Helpers/BindableBase.cs

```
namespace WPFApp.Helpers;

public class BindableBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected void RaisePropertyChanged([CallerMemberName] string propertyName = null)
```

```
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }

    protected void SetProperty<T>(ref T dest, T val, [CallerMemberName] string
propertyName = null)
        where T : IComparable<T>
    {
        if (dest.CompareTo(val) != 0)
        {
            dest = val;
            RaisePropertyChanged(propertyName);
        }
    }
}
```

WpfApp/Model/Sentence.cs

```
public class Sentence : BindableBase
{
    private string _nameOfGame = string.Empty;

    public string NameOfGame
    {
        get => _nameOfGame;
        set => SetProperty(ref _nameOfGame, value);
    }

    // ...
}
```

8.3. OnPropertyChanged

ViewModel.cs

```
public class GameViewModel : INotifyPropertyChanged
{
    private Supermarket _supermarket;
    public Supermarket Supermarket
    {
        get => _supermarket;
        set
        {
            _supermarket = value;
            OnPropertyChanged();
        }
    }
```

```
    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual void OnPropertyChanged([CallerMemberName] string propertyName =
null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

8.4. Directories

```
private static string BaseDirectory =>
Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
private string _filename = $"{BaseDirectory}\\game.txt";
```

8.5. HttpClientFactory

```
public SudokuSolveService(IHttpClientFactory httpFactory)
{
    _httpFactory = httpFactory;
}

private readonly IHttpClientFactory _httpFactory;

public HttpClient Http => _httpFactory.CreateClient("leocloud");


private string ToFirstQuery(IEnumerable<string> sudoku)
{
    return "?sudoku=" + string.Join("&sudoku=", sudoku);
}

public async Task<SudokuSolveResult?> Solve(IEnumerable<string> sudoku)
{
    try
    {
        return await Http.GetFromJsonAsync<SudokuSolveResult>("Sudoku" +
ToFirstQuery(sudoku));
    }
    catch (Exception){ return null;}
}

public async Task<int> GetSolutionCount(IEnumerable<string> sudoku)
{
    try
    {
        var query         = ToFirstQuery(sudoku);
        var solutionCount = await Http.GetFromJsonAsync<int>("Sudoku/solutioncount" +
```

```
query);
        return solutionCount;
    }
    catch (Exception){}
    return -1;
}
```

8.6. Sodoku Field Creation

```
public MainWindow(MainWindowViewModel vm)
{
    vm.Controller = new WindowNavigator(this);
    InitializeComponent();
    DataContext = vm;

    for (int row = 0; row < 9; row++)
    {
        var c = new Label
        {
            Content = row+1,
            VerticalContentAlignment = VerticalAlignment.Center
        };
        c.SetValue(Grid.RowProperty,    row + 1);
        c.SetValue(Grid.ColumnProperty, 0);

        SudokuGrid.Children.Add(c);
    }

    for (int col = 0; col < 9; col++)
    {
        var c = new Label
        {
            Content                    = (char) (col + 'A'),
            HorizontalContentAlignment = HorizontalAlignment.Center
        };
        c.SetValue(Grid.RowProperty,    0);
        c.SetValue(Grid.ColumnProperty, col+1);

        SudokuGrid.Children.Add(c);
    }


    for (int row = 0; row < 9; row++)
    {
        for (int col = 0; col < 9; col++)
        {
            var c = new SudokuCell
            {
                Command          = vm.CellCommand,
```

```
                CommandParameter = vm.GetCell(row, col),
                Value            = vm.GetCell(row, col)
            };
            c.SetValue(Grid.RowProperty,    row+1);
            c.SetValue(Grid.ColumnProperty, col+1);

            SudokuGrid.Children.Add(c);
        }
    }

    Loaded += async (v, e) =>
    {
        await (DataContext as MainWindowViewModel)!
            .LoadDataAsync();
    };
}
```

8.7. ExceptBy

```
var newQuestions = questionsInCsv
    .ExceptBy(exam.ExamQuestions!.Select(m => m.Number), m => m.Number)
    .ToList();

var inDb = await _uow.Exam.GetExamExamineeAsync();
return files.ExceptBy(inDb.Select(e => (e.ExamDate, e.ExamName, e.ExamineeName)), e =>
(e.ExamDate, e.ExamName, e.ExamineeName)).ToList();
```

8.8. Date Parse

```
string dateString = "20231124";
DateOnly date = DateOnly.ParseExact(dateString, "yyyyMMdd",
CultureInfo.InvariantCulture);

//
Duration = double.Parse(parts[3], CultureInfo.InvariantCulture),
Date = DateOnly.FromDateTime(DateTime.ParseExact(parts[1], "dd.MM.yyyy",
CultureInfo.InvariantCulture)),
```

8.9 Select with Index

```
return bookings.Select((b, index) => new BookingSummary
        {
            No = index,
            End = b.CheckOutDate,
            Start = b.CheckInDate,
            Name = b.Guest.Name,
```

```
            BookingId = b.Id
        }).ToList();
```

8.10 SelectMany

```
var uniqueRoomSpecials = csvFile
    .SelectMany(l => l.RoomSpecials
        .Split(',')
        .Where(rs => roomSpecialInDb.All(rsDb => rsDb.Special != rs) &&
!String.IsNullOrEmpty(rs))
        .Select(rs => new RoomSpecial()
        {
            Special = rs
        }))
    .GroupBy(rs => rs.Special)
    .Select(g => g.First())
    .ToList();

await uow.RoomSpecial.AddRangeAsync(uniqueRoomSpecials);
```

8.11 String.Join

```
public async Task<IList<RoomSummary>> GetRoomSummaryListAsync()
{
    return await DbSet.Select(n => new RoomSummary()
    {
        Id = n.Id,
        Type = n.RoomType.Type,
        BookingCount = n.Bookings.Count(),
        RoomNumber = n.RoomNumber,
        Specials = string.Join(", ", n.Specials.Select(s => s.Special))
    }).ToListAsync();
}
```

4. Other

1. File Functions

```
// Prüft, ob eine Datei existiert
if (File.Exists("datei.txt"))
{
    Console.WriteLine("Datei existiert.");
}

// Liest den gesamten Inhalt einer Datei als String
```

```
string content = File.ReadAllText("datei.txt");

// Schreibt einen String in eine Datei
File.WriteAllText("datei.txt", "Das ist der Inhalt der Datei.");

// Fügt Text an eine Datei an (erstellt sie, falls nicht vorhanden)
File.AppendAllText("datei.txt", "Zusätzlicher Inhalt.");

// Liest alle Zeilen einer Datei als Array
string[] rows = File.ReadAllLines("datei.txt");

// Schreibt ein Array von Zeilen in eine Datei
File.WriteAllLines("datei.txt", new string[] { "Zeile 1", "Zeile 2" });

// Löscht eine Datei
File.Delete("datei.txt");

// Kopiert eine Datei
File.Copy("quelle.txt", "ziel.txt");

// Verschiebt oder benennt eine Datei um
File.Move("alteDatei.txt", "neueDatei.txt");

// Öffnet einen Stream zum Lesen einer Datei
using (FileStream fs = File.OpenRead("datei.txt"))
{
    // Datei wird nur gelesen
}

// Öffnet einen Stream zum Schreiben in eine Datei
using (FileStream fs = File.OpenWrite("datei.txt"))
{
    // Datei wird nur beschrieben
}

// Gibt die Größe einer Datei in Bytes zurück
long size = new FileInfo("datei.txt").Length;

// Erzeugt eine leere Datei (falls sie nicht existiert)
File.Create("neueDatei.txt").Dispose();
```

5. Specifics

5.1. Zahlen aus String

```
static void Main()
{
    string input = "abscdef123dsdf";
```

```
    int number = ExtractNumber(input);
    Console.WriteLine(number); // Ausgabe: 123
}

static int ExtractNumber(string input)
{
    int result = 0;
    foreach (char c in input)
    {
        if (char.IsDigit(c))
        {
            result = result * 10 + (c - '0');
        }
    }
    return result;
}
```

5.2. Form nicht erkannt (Angular)

```
standalone: true,
imports: [FormsModule]
```

5.3. ToDto function

```
public record StationDto(
    int           Id,
    string?       Remark,
    IList<string> Infrastructures,
);

Station ToEntity(StationDto dto)
{
    return new Station()
    {
        Id          = dto.Id,
        Remark      = dto.Remark,
    };
}

StationDto? ToDto(Station? entity)
{
    if (entity is null) return null;

    return new StationDto(
        entity.Id,
        entity.Remark,
        entity.City?.Name ?? string.Empty,
```

```
        entity.Infrastructures?.Select(x => x.Name).ToList() ?? new List<string>()
    );
}

IList<StationDto>? ToDto(IList<Station>? list)
{
    if (list is null) return null;
    return list.Select(x => ToDto(x)!).ToList();
}
```