pkg VMSApi.Mappers

PostMapperFoodPurchase

+ «property» Date: DateTime {Annotation = Required}
+ «property» Amount: float {Annotation = Required}
+ «property» GroupNumber: string {Annotation = Required}

«interface»
IMappings

MapToMeasurePoint (mapperMeasurePoint : PostMapperMeasurePoint): MeasurePoint
MapToFoodPurchase (mapperFoodPurchase : PostMapperFoodPurchase): FoodPurchase

AutoMapperProfile

+ «create» AutoMapperProfile ()

Mappings

+ MapToMeasurePoint (mapperMeasurePoint : PostMapperMeasurePoint): MeasurePoint
+ MapToFoodPurchase (mapperFoodPurchase : PostMapperFoodPurchase): FoodPurchase

PostMapperMeasurePoint

+ «property» Date: DateTime {Annotation = Required}
+ «property» Weight: float {Annotation = Required}
+ «property» Amount: int
+ «property» GroupNumber: string {Annotation = Required}

pkg VMSApi.CustomAttributes

AllowedUserRoles

- _allowedValues: string[]
+ «create» AllowedUserRoles (allowedValues : string[])
# IsValid (value : object, validationContext : ValidationContext): ValidationResult

pkg VMSApi.Exceptions

RolesException

+ «create» RolesException (message : string)

pkg VMSApi.Controllers

PigGroupController

- _pigGroupRepo: IPigGroupRepo {readOnly}
+ «create» PigGroupController (pigGroupRepo : IPigGroupRepo)
+ «async» GetAll (): Task<IActionResult> {Annotation = Route("groups")} {Annotation = HttpGet}
+ «async» GetById (id : Guid): Task<IActionResult>
+ «async» GetByDate (startDate : DateTime?, endDate : DateTime?): Task<IActionResult>
+ «async» Create (pigGroup : PigGroup): Task<IActionResult>
+ «async» Update (id : Guid, existingGroup : PigGroup): Task<IActionResult> {Annotation = Route("groups/{id}")} {Annotation = HttpPut}

MeasurePointController

- _measurePointRepo: IMeasurePointRepo {readOnly}
- _mappings: IMappings {readOnly}
+ «create» MeasurePointController (measurePointReo : IMeasurePointRepo, mappings : IMappings)
+ «async» GetAll (): Task<IActionResult> {Annotation = Route("Points")} {Annotation = HttpGet}
+ «async» GetById (id : Guid): Task<IActionResult>
+ «async» Create (newMapperPoint : PostMapperMeasurePoint): Task<IActionResult>
+ «async» Update (id : Guid, updatedMeasurePoint : MeasurePoint): Task<IActionResult> {Annotation = Route("Points/{id}")} {Annotation = HttpPut}

TaskItemsController

- _repository: ITaskItemRepository {readOnly}
- _mapper: IMapper {readOnly}
+ «create» TaskItemsController (repository : ITaskItemRepository, mapper : IMapper)
+ «async» GetAsync (): Task<IActionResult>
+ «async» GetByIdAsync (id : Guid): Task<IActionResult>
+ «async» PostAsync (taskItemModel : PostTaskItemModel): Task<IActionResult>
+ «async» PutAsync (id : Guid, taskItem : PostTaskItemModel): Task<IActionResult> {Annotation = HttpPut("{id}")}
+ «async» AssignUserToTaskPostAsync (taskid : Guid, model : AssignTaskToUserModel): Task<IActionResult>
+ «async» Delete (id : Guid): Task<IActionResult> {Annotation = HttpDelete("{id:guid}")}

AccountsController

- _mapper: IMapper {readOnly}
- _userManager: UserManager<User> {readOnly}
- _tokenGenerator: ITokenGenerator {readOnly}
+ «create» AccountsController (mapper : IMapper, userManager : UserManager<User>, tokenGenerator : ITokenGenerator)
+ «async» Register (userModel : UserRegistrationModel): Task<ActionResult>
+ «async» GetAllAsync (): Task<IActionResult> {Annotation = ExcludeFromCodeCoverage} {Annotation = HttpGet}
+ «async» LoginEndpoint (userModel : UserLoginModel): Task<IActionResult> {Annotation = HttpPost("Login", Name = "LoginEndpoint")}
+ «async» DeleteUserEndpoint (userid : Guid): Task<IActionResult> {Annotation = Route("Delete/{userid:guid}", Name = "DeleteUserById")} {Annotation = HttpDelete} {Annotation = Authorize(Roles = "Manager,Administrator")}
+ «async» AssignRolesToUser (userid : Guid, model : ChangeRolesModel): Task<IActionResult> {Annotation = ExcludeFromCodeCoverage} {Annotation = Route("Roles/{userid:guid}")} {Annotation = HttpPost} {Annotation = Authorize(Roles = "Administrator")}
+ «async» UpdateUser (userid : Guid, model : UserUpdateModel): Task<IActionResult> {Annotation = Route("UpdateAsync/{userid:guid}")} {Annotation = Authorize(Roles = "Administrator,Manager")} {Annotation = HttpPut}
+ «async» UpdatePassword (userid : Guid, model : PasswordResetModel): Task<IActionResult> {Annotation = Route("{userid:guid}/ChangePassword")} {Annotation = Authorize} {Annotation = HttpPost}
- «async» UpdateUserProps (user : User, model : UserUpdateModel): Task {Annotation = ExcludeFromCodeCoverage}
- ValidateRoles (modelRoles : List<string>) {Annotation = ExcludeFromCodeCoverage}
- «async» UpdateUserRoles (user : User, newRoles : List<string>): Task {Annotation = ExcludeFromCodeCoverage}

FoodPurchaseController

- _foodPurchaseRepo: IFoodPurchasesRepo {readOnly}
- _mappings: IMappings {readOnly}
+ «create» FoodPurchaseController (foodPurchasesRepo : IFoodPurchasesRepo, mappings : IMappings)
+ «async» GetAll (): Task<IActionResult> {Annotation = Route("Purchases")} {Annotation = HttpGet}
+ «async» GetById (id : Guid): Task<IActionResult>
+ «async» CreateAsync (newMapperPurchase : PostMapperFoodPurchase): Task<IActionResult>
+ «async» Update (id : Guid, newPurchase : FoodPurchase): Task<IActionResult> {Annotation = Route("Purchases/{id}")} {Annotation = HttpPut}

pkg VMSApi.Services

«interface»
ITokenGenerator

GetTokenAsync (user : User): Task<LoginJwtTokenDto>

JwtBearerTokenGenerator

- _userManager: UserManager<User> {readOnly}
- _jwtSettings: IConfigurationSection {readOnly}
+ «create» JwtBearerTokenGenerator (userManager : UserManager<User>, jwtSettings : IConfiguration)
+ «async» GetTokenAsync (user : User): Task<LoginJwtTokenDto>
- GetSigningCredentials (): SigningCredentials
- GenerateTokenOptions (signingCredentials : SigningCredentials, claims : List<Claim>): JwtSecurityToken
- «async» GetClaims (user : User): Task<List<Claim>>