Namespace BankBackend.Controllers

Classes

AccountController

UsersController

Class AccountController

Namespace: BankBackend.Controllers

Assembly: BankBackend.dll

```
[ApiController]
[Route("[controller]")]
public class AccountController : ControllerBase
```

Inheritance

object ← ControllerBase ← AccountController

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) , ControllerBase.Content(string) , ControllerBase.Content(string, string) , ControllerBase.Content(string, string, Encoding) , ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() , ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) , ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) , ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) , ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) , ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() , ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) , ControllerBase.RedirectToAction(string, string) , ControllerBase.RedirectToAction(string, string, object) , ControllerBase.RedirectToAction(string, string, string) , ControllerBase.RedirectToAction(string, string, object, string) , ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) , ControllerBase.RedirectToActionPermanent(string) , ControllerBase.RedirectToActionPermanent(string, object) , ControllerBase.RedirectToActionPermanent(string, string) , ControllerBase.RedirectToActionPermanent(string, string, string) , ControllerBase.RedirectToActionPermanent(string, string, object) , ControllerBase.RedirectToActionPermanent(string, string, object, string) , ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) , ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,

ControllerBase.RedirectToRoute(string, object) ,
ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) ,
ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) ,
ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) ,
ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,

ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) ,
ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) ,
ControllerBase.NotFound() , ControllerBase.NotFound(object) ,
ControllerBase.BadRequest() , ControllerBase.BadRequest(object) ,
ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) ,
ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary)
,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) ,
ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) ,
ControllerBase.Accepted(string) , ControllerBase.Accepted(string, object) ,
ControllerBase.Accepted(Uri, object) , ControllerBase.AcceptedAtAction(string) ,
ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,

ControllerBase.AcceptedAtRoute(string, object) ,
ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) ,
ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) ,
ControllerBase.Forbid() , ControllerBase.Forbid(params string[]) ,
ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) ,
ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) ,
ControllerBase.SignOut(params string[]) ,
ControllerBase.SignOut(AuthenticationProperties, params string[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync(object, Type, string) ,
ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryValidateModel(object) ,
ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext ,
ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData ,
ControllerBase.ModelState , ControllerBase.ControllerContext ,
ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory ,
ControllerBase.Url , ControllerBase.ObjectValidator ,
ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty ,
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() ,
object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) ,
object.ToString()

Constructors

AccountController(ILogger<AccountController>, IBankService)

```
public AccountController(ILogger<AccountController> logger,
IBankService bankService)
```

Parameters

`logger` ILogger<AccountController>

`bankService` IBankService

Methods

GetAllAccounts()

```
[HttpGet("")]
public List<Account> GetAllAccounts()
```

Returns

List<Account>

GetTransactionsByAccountId(int)

```
[HttpGet("{id}")]
public List<Transaction>? GetTransactionsByAccountId(int id)
```

Parameters

`id` int

Returns

List<Transaction>

PostAccount(Account)

```
[HttpPost("")]
public Account PostAccount(Account account)
```

Parameters

`account` Account

Returns

Account

Class UsersController

Namespace: BankBackend.Controllers

Assembly: BankBackend.dll

```
[ApiController]
[Route("[controller]")]
public class UsersController : ControllerBase
```

Inheritance

object ← ControllerBase ← UsersController

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) , ControllerBase.Content(string) , ControllerBase.Content(string, string) , ControllerBase.Content(string, string, Encoding) , ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() , ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) , ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) , ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) , ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) , ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() , ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) , ControllerBase.RedirectToAction(string, string) , ControllerBase.RedirectToAction(string, string, object) , ControllerBase.RedirectToAction(string, string, string) , ControllerBase.RedirectToAction(string, string, object, string) , ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) , ControllerBase.RedirectToActionPermanent(string) , ControllerBase.RedirectToActionPermanent(string, object) , ControllerBase.RedirectToActionPermanent(string, string) , ControllerBase.RedirectToActionPermanent(string, string, string) , ControllerBase.RedirectToActionPermanent(string, string, object) , ControllerBase.RedirectToActionPermanent(string, string, object, string) , ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) , ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,

ControllerBase.RedirectToRoute(string, object) ,
ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) ,
ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) ,
ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) ,
ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,

ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) ,
ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) ,
ControllerBase.NotFound() , ControllerBase.NotFound(object) ,
ControllerBase.BadRequest() , ControllerBase.BadRequest(object) ,
ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) ,
ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary)
,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) ,
ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) ,
ControllerBase.Accepted(string) , ControllerBase.Accepted(string, object) ,
ControllerBase.Accepted(Uri, object) , ControllerBase.AcceptedAtAction(string) ,
ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,

ControllerBase.AcceptedAtRoute(string, object) ,
ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) ,
ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) ,
ControllerBase.Forbid() , ControllerBase.Forbid(params string[]) ,
ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) ,
ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) ,
ControllerBase.SignOut(params string[]) ,
ControllerBase.SignOut(AuthenticationProperties, params string[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) ,
ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryUpdateModelAsync(object, Type, string) ,
ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) ,
ControllerBase.TryValidateModel(object) ,
ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext ,
ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData ,
ControllerBase.ModelState , ControllerBase.ControllerContext ,
ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory ,
ControllerBase.Url , ControllerBase.ObjectValidator ,
ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty ,
object.Equals(object) , object.Equals(object, object) , object.GetHashCode() ,
object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) ,
object.ToString()

Constructors

UsersController(ILogger<UsersController>, IBankService)

```
public UsersController(ILogger<UsersController> logger, IBankService bankService)
```

Parameters

`logger` ILogger<UsersController>

`bankService` IBankService

Methods

AddAccountToUserById(int, int, int)

```
[HttpPatch("{userId}/add/{addedUser}")]
public Account? AddAccountToUserById(int userId, int addedUser, int accountId)
```

Parameters

`userId` int

`addedUser` int

`accountId` int

Returns

Account

Deposit(int, int, double)

```
[HttpPatch("{userId}/deposit")]
public Transaction? Deposit(int userId, int accountId, double amount)
```

Parameters

`userId` int

`accountId` int

`amount` double

Returns

Transaction

GetAccountsByUserId(int)

```
[HttpGet("{id}/accounts")]
public List<Account>? GetAccountsByUserId(int id)
```

Parameters

`id` int

Returns

List<Account>

GetAllUsers()

```
[HttpGet("")]
public List<User> GetAllUsers()
```

Returns

List<User>

GetTransactionsByUserId(int)

```
[HttpGet("{id}/transactions")]
public List<Transaction>? GetTransactionsByUserId(int id)
```

Parameters

`id` int

Returns

List<Transaction>

GetUsersById(int)

```
[HttpGet("{userId}")]
public User? GetUsersById(int userId)
```

Parameters

`userId` int

Returns

User

Login(User)

```
[HttpPost("login")]
public User? Login(User user)
```

Parameters

`user` User

Returns

User

PostUser(User)

```
[HttpPost("")]
public User PostUser(User user)
```

Parameters

`user` User

Returns

User

RemoveUserFromAccountById(int, int)

```
[HttpPatch("{userId}/remove")]
public Account? RemoveUserFromAccountById(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

Account

Transfer(int, int, int, double)

```
[HttpPatch("{userId}/transfer")]
public Transaction? Transfer(int userId, int fromAccountId, int toAccountId,
double amount)
```

Parameters

`userId` int

`fromAccountId` int

`toAccountId` int

`amount` double

Returns

Transaction

UpdateUserInfo(int, User)

```
[HttpPatch("{userId}")]
public User? UpdateUserInfo(int userId, User user)
```

Parameters

`userId` int

`user` User

Returns

User

Withdraw(int, int, double)

```
[HttpPatch("{userId}/withdraw")]
public Transaction? Withdraw(int userId, int accountId, double amount)
```

Parameters

`userId` int

`accountId` int

`amount` double

Returns

Transaction

Namespace BankBackend.Models

Classes

Account

Transaction

User

Enums

AccountType

different types of accounts like savings or checkings

Class Account

Namespace: BankBackend.Models

Assembly: BankBackend.dll

```
public class Account
```

Inheritance

object ← Account

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Account()

```
public Account()
```

Properties

AccountId

```
[Key]
public int AccountId { get; set; }
```

Property Value

int

Balance

```
public double Balance { get; set; }
```

Property Value

double

PrimaryUserId

```
[ForeignKey("UserId")]
public int PrimaryUserId { get; set; }
```

Property Value

int

Type

```
public AccountType Type { get; set; }
```

Property Value

AccountType

Users

```
public List<User> Users { get; set; }
```

Property Value

List<User>

Enum AccountType

Namespace: BankBackend.Models

Assembly: BankBackend.dll

different types of accounts like savings or checkings

```
public enum AccountType
```

Fields

CHECKING = 1

CLOWN = 2

SAVINGS = 0

Class Transaction

Namespace: BankBackend.Models

Assembly: BankBackend.dll

```
public class Transaction
```

Inheritance

object ← Transaction

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Transaction()

```
public Transaction()
```

Transaction(Account, Account, double)

```
public Transaction(Account fromAccount, Account toAccount, double amount)
```

Parameters

`fromAccount` Account

`toAccount` Account

`amount` double

Transaction(Account, double)

```
public Transaction(Account account, double amount)
```

Parameters

`account` Account

`amount` double

Properties

Amount

```
public double Amount { get; set; }
```

Property Value

double

FromAccount

```
public Account FromAccount { get; set; }
```

Property Value

Account

Time

```
public DateTime Time { get; set; }
```

Property Value

DateTime

ToAccount

```
public Account? ToAccount { get; set; }
```

Property Value

Account

TransactionId

```
[Key]
public int TransactionId { get; set; }
```

Property Value

int

Class User

Namespace: BankBackend.Models

Assembly: BankBackend.dll

```
[Index("Username", new string[] { }, IsUnique = true)]
public class User
```

Inheritance

object ← User

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

User()

```
public User()
```

User(string, string, string)

```
public User(string username, string password, string name)
```

Parameters

`username` string

`password` string

`name` string

Properties

Accounts

```
public List<Account> Accounts { get; set; }
```

Property Value

List<Account>

Name

```
public string Name { get; set; }
```

Property Value

string

Password

```
public string Password { get; set; }
```

Property Value

string

UserId

```
[Key]
public int UserId { get; set; }
```

Property Value

int

Username

```
public string Username { get; set; }
```

Property Value

string

Namespace BankBackend.Repository

Classes

BankContext

BankRepository

Interfaces

IBankRepository

Class BankContext

Namespace: BankBackend.Repository

Assembly: BankBackend.dll

```
public class BankContext : DbContext, IInfrastructure<IServiceProvider>,
IDbContextDependencies, IDbSetCache, IDbContextPoolable, IResettableService,
IDisposable, IAsyncDisposable
```

Inheritance

object ← DbContext ← BankContext

Implements

IInfrastructure<IServiceProvider>, IDbContextDependencies, IDbSetCache, IDbContextPoolable, IResettableService, IDisposable, IAsyncDisposable

Inherited Members

DbContext.Set<TEntity>() , DbContext.Set<TEntity>(string) , DbContext.OnConfiguring(DbContextOptionsBuilder) , DbContext.ConfigureConventions(ModelConfigurationBuilder) , DbContext.OnModelCreating(ModelBuilder) , DbContext.SaveChanges() , DbContext.SaveChanges(bool) , DbContext.SaveChangesAsync(CancellationToken) , DbContext.SaveChangesAsync(bool, CancellationToken) , DbContext.Dispose() , DbContext.DisposeAsync() , DbContext.Entry<TEntity>(TEntity) , DbContext.Entry(object) , DbContext.Add<TEntity>(TEntity) , DbContext.AddAsync<TEntity>(TEntity, CancellationToken) , DbContext.Attach<TEntity>(TEntity) , DbContext.Update<TEntity>(TEntity) , DbContext.Remove<TEntity>(TEntity) , DbContext.Add(object) , DbContext.AddAsync(object, CancellationToken) , DbContext.Attach(object) , DbContext.Update(object) , DbContext.Remove(object) , DbContext.AddRange(params object[]) , DbContext.AddRangeAsync(params object[]) , DbContext.AttachRange(params object[]) , DbContext.UpdateRange(params object[]) , DbContext.RemoveRange(params object[]) , DbContext.AddRange(IEnumerable<object>) , DbContext.AddRangeAsync(IEnumerable<object>, CancellationToken) , DbContext.AttachRange(IEnumerable<object>) , DbContext.UpdateRange(IEnumerable<object>) , DbContext.RemoveRange(IEnumerable<object>) , DbContext.Find(Type, params object[]) , DbContext.FindAsync(Type, params object[]) ,

DbContext.FindAsync(Type, object[], CancellationToken) , DbContext.Find<TEntity>(params object[]) , DbContext.FindAsync<TEntity>(params object[]) , DbContext.FindAsync<TEntity>(object[], CancellationToken) , DbContext.FromExpression<TResult>(Expression<Func<IQueryable<TResult>>>) , DbContext.Database , DbContext.ChangeTracker , DbContext.Model , DbContext.ContextId , DbContext.SavingChanges , DbContext.SavedChanges , DbContext.SaveChangesFailed , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

BankContext(DbContextOptions<BankContext>)

```
public BankContext(DbContextOptions<BankContext> options)
```

Parameters

`options` DbContextOptions<BankContext>

Properties

Accounts

```
public DbSet<Account> Accounts { get; }
```

Property Value

DbSet<Account>

Transactions

```
public DbSet<Transaction> Transactions { get; }
```

Property Value

DbSet<Transaction>

Users

```
public DbSet<User> Users { get; }
```

Property Value

DbSet<User>

Class BankRepository

Namespace: BankBackend.Repository

Assembly: BankBackend.dll

```
public class BankRepository : IBankRepository
```

Inheritance

object ← BankRepository

Implements

IBankRepository

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

BankRepository(BankContext)

constructor for dependency injection

```
public BankRepository(BankContext bankContext)
```

Parameters

`bankContext` BankContext

BankRepository(string)

```
public BankRepository(string connectionString)
```

Parameters

`connectionString` string

Methods

AddAccountToUser(int, int)

adds account with `accountId` to the account list of the user with `userId`

```
public User? AddAccountToUser(int accountId, int userId)
```

Parameters

`accountId` int

`userId` int

Returns

User

the updated user with the added account, null if the user does not exist

AddUserToAccount(int, int)

adds user with `userId` to the account list of the account with `accountId`

```
public Account? AddUserToAccount(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

Account

the updated account with the new user, null if account does not exist or if user does not exist

CreateAccount(Account)

uploads the account to database

```
public Account CreateAccount(Account account)
```

Parameters

`account` Account

Returns

Account

the account that was just created

CreateTransaction(Transaction)

uploads the transaction to database

```
public Transaction CreateTransaction(Transaction transaction)
```

Parameters

`transaction` Transaction

Returns

Transaction

the transaction that was just created

CreateUser(User)

uploads the user to database

```
public User CreateUser(User user)
```

Parameters

`user` User

Returns

User

the user that was just created

DeleteAccountById(int)

deletes the account with `accountId`

```
public Account? DeleteAccountById(int accountId)
```

Parameters

`accountId` int

Returns

Account

the account that was just deleted, null if the account does not exist

DeleteAccountUserByUserId(int, int)

deletes the user with `userId` from the account with `accountId`

```
public User? DeleteAccountUserByUserId(int accountId, int userId)
```

Parameters

`accountId` int

`userId` int

Returns

User

the user that was just deleted, null if the account does not exist, or if the account does not have an user with `userId`

DeleteTransactionByTransactionId(int)

deletes the transaction with `transactionId`

```
public Transaction? DeleteTransactionByTransactionId(int transactionId)
```

Parameters

`transactionId` int

Returns

Transaction

the transaction that was just deleted, null if the transaction does not exist

DeleteUserAccountByAccountId(int, int)

deletes the account with `accountId` from the user with `userId`

```
public Account? DeleteUserAccountByAccountId(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

Account

the account that was just deleted, null if user does not exist, or the user does not have an account with `accountId`

DeleteUserById(int)

deletes the user with `userId`

```
public User? DeleteUserById(int userId)
```

Parameters

`userId` int

Returns

User

the user that was just deleted, null if the user does not exist

GetAccountByAccountId(int)

find an account with `accountId`

```
public Account? GetAccountByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

Account

the account with `accountId`, null if account does not exist

GetAccountsByUserId(int)

find all the accounts of the user with `userId`

```
public List<Account>? GetAccountsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Account>

a list containing all the accounts, null if user does not exist

GetAllAccounts()

find all existing accounts in the database

```
public List<Account> GetAllAccounts()
```

Returns

List<Account>

list containing all existing accounts, empty list if non exists

GetAllTransactions()

find all existing transactions in the database

```
public List<Transaction> GetAllTransactions()
```

Returns

List<Transaction>

list containing all existing transactions, empty list if non exists

GetAllUsers()

find all existing users in the database

```
public List<User> GetAllUsers()
```

Returns

List <User>

a list containing all existing users, empty list if non exists

GetPrimaryAccountsByUserId(int)

find all the accounts that the user with `accountId` is a primary user

```
public List<Account>? GetPrimaryAccountsByUserId(int userId)
```

Parameters

`userId` int

Returns

List <Account>

a list containing all the primary accounts, null if user does not exist

GetTransactionByTransactionId(int)

find a transaction with `transactionId`

```
public Transaction? GetTransactionByTransactionId(int transactionId)
```

Parameters

`transactionId` int

Returns

Transaction

the transaction with `transactionId`, null if transaction does not exist

GetTransactionsByFromAccountId(int)

```
public List<Transaction> GetTransactionsByFromAccountId(int fromAccountId)
```

Parameters

`fromAccountId` int

Returns

List<Transaction>

GetTransactionsByToAccountId(int)

```
public List<Transaction> GetTransactionsByToAccountId(int toAccountId)
```

Parameters

`toAccountId` int

Returns

List<Transaction>

GetUserByUserId(int)

find a user with `userId`

```
public User? GetUserByUserId(int userId)
```

Parameters

`userId` int

Returns

User

the user with the Id, null if user does not exist

GetUserByUsername(string)

find a user with `username`

```
public User? GetUserByUsername(string username)
```

Parameters

`username` string

Returns

User

the user with the username, null if user does not exist

GetUsersByAccountId(int)

find all the users of the account with `accountId`

```
public List<User>? GetUsersByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

List<User>

a list containing all the users, null if account does not exist

UpdateBalance(int, double)

updates the balance of the account with `accountId` to `balance`

```
public Account? UpdateBalance(int accountId, double balance)
```

Parameters

`accountId` int

`balance` double

Returns

Account

the updated account with the new balance, null if the account does not exist

UpdatePassword(int, string)

updates the password of user with `userId` to `password`

```
public User? UpdatePassword(int userId, string password)
```

Parameters

`userId` int

`password` string

Returns

User

the updated user with the new password, null if the user does not exist

UpdatePrimaryUser(int, int)

replaces the primary user of the account with `accountId` with `userId`

```
public Account? UpdatePrimaryUser(int accountId, int userId)
```

Parameters

`accountId` int

`userId` int

Returns

Account

the updated account with the new primary user, null if account doesnot exist or if user does not exist

UpdateUsername(int, string)

updates the name of the user with `userId` to `name`

```
public User? UpdateUsername(int userId, string username)
```

Parameters

`userId` int

`username` string

Returns

User

the updated user with the new username, null if the user does not exist

Interface IBankRepository

Namespace: BankBackend.Repository

Assembly: BankBackend.dll

```
public interface IBankRepository
```

Methods

AddAccountToUser(int, int)

adds account with `accountId` to the account list of the user with `userId`

```
User? AddAccountToUser(int accountId, int userId)
```

Parameters

`accountId` int

`userId` int

Returns

User

the updated user with the added account, null if the user does not exist

AddUserToAccount(int, int)

adds user with `userId` to the account list of the account with `accountId`

```
Account? AddUserToAccount(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

Account

the updated account with the new user, null if account does not exist or if user does not exist

CreateAccount(Account)

uploads the account to database

```
Account CreateAccount(Account account)
```

Parameters

`account` Account

Returns

Account

the account that was just created

CreateTransaction(Transaction)

uploads the transaction to database

```
Transaction CreateTransaction(Transaction transaction)
```

Parameters

`transaction` Transaction

Returns

Transaction

the transaction that was just created

CreateUser(User)

uploads the user to database

```
User CreateUser(User user)
```

Parameters

`user` User

Returns

User

the user that was just created

DeleteAccountById(int)

deletes the account with `accountId`

```
Account? DeleteAccountById(int accountId)
```

Parameters

`accountId` int

Returns

Account

the account that was just deleted, null if the account does not exist

DeleteAccountUserByUserId(int, int)

deletes the user with `userId` from the account with `accountId`

```
User? DeleteAccountUserByUserId(int accountId, int userId)
```

Parameters

`accountId` int

`userId` int

Returns

User

the user that was just deleted, null if the account does not exist, or if the account does not have an user with `userId`

DeleteTransactionByTransactionId(int)

deletes the transaction with `transactionId`

```
Transaction? DeleteTransactionByTransactionId(int transactionId)
```

Parameters

`transactionId` int

Returns

Transaction

the transaction that was just deleted, null if the transaction does not exist

DeleteUserAccountByAccountId(int, int)

deletes the account with `accountId` from the user with `userId`

```
Account? DeleteUserAccountByAccountId(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

Account

the account that was just deleted, null if user does not exist, or the user does not have an account with `accountId`

DeleteUserById(int)

deletes the user with `userId`

```
User? DeleteUserById(int userId)
```

Parameters

`userId` int

Returns

User

the user that was just deleted, null if the user does not exist

GetAccountByAccountId(int)

find an account with `accountId`

```
Account? GetAccountByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

Account

the account with `accountId`, null if account does not exist

GetAccountsByUserId(int)

find all the accounts of the user with `userId`

```
List<Account>? GetAccountsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Account>

a list containing all the accounts, null if user does not exist

GetAllAccounts()

find all existing accounts in the database

```
List<Account> GetAllAccounts()
```

Returns

List<Account>

list containing all existing accounts, empty list if non exists

GetAllTransactions()

find all existing transactions in the database

```
List<Transaction> GetAllTransactions()
```

Returns

List<Transaction>

list containing all existing transactions, empty list if non exists

GetAllUsers()

find all existing users in the database

```
List<User> GetAllUsers()
```

Returns

List<User>

a list containing all existing users, empty list if non exists

GetPrimaryAccountsByUserId(int)

find all the accounts that the user with `accountId` is a primary user

```
List<Account>? GetPrimaryAccountsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Account>

a list containing all the primary accounts, null if user does not exist

GetTransactionByTransactionId(int)

find a transaction with `transactionId`

```
Transaction? GetTransactionByTransactionId(int transactionId)
```

Parameters

`transactionId` int

Returns

Transaction

the transaction with `transactionId`, null if transaction does not exist

GetTransactionsByFromAccountId(int)

```
List<Transaction> GetTransactionsByFromAccountId(int fromAccountId)
```

Parameters

`fromAccountId` int

Returns

List<Transaction>

GetTransactionsByToAccountId(int)

```
List<Transaction> GetTransactionsByToAccountId(int toAccountId)
```

Parameters

`toAccountId` int

Returns

List<Transaction>

GetUserByUserId(int)

find a user with `userId`

```
User? GetUserByUserId(int userId)
```

Parameters

`userId` int

Returns

User

the user with the Id, null if user does not exist

GetUserByUsername(string)

find a user with `username`

```
User? GetUserByUsername(string username)
```

Parameters

`username` string

Returns

User

the user with the username, null if user does not exist

GetUsersByAccountId(int)

find all the users of the account with `accountId`

```
List<User>? GetUsersByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

List<User>

a list containing all the users, null if account does not exist

UpdateBalance(int, double)

updates the balance of the account with `accountId` to `balance`

```
Account? UpdateBalance(int accountId, double balance)
```

Parameters

`accountId` int

`balance` double

Returns

Account

the updated account with the new balance, null if the account does not exist

UpdatePassword(int, string)

updates the password of user with `userId` to `password`

```
User? UpdatePassword(int userId, string password)
```

Parameters

`userId` int

`password` string

Returns

User

the updated user with the new password, null if the user does not exist

UpdatePrimaryUser(int, int)

replaces the primary user of the account with `accountId` with `userId`

```
Account? UpdatePrimaryUser(int accountId, int userId)
```

Parameters

`accountId` int

`userId` int

Returns

Account

the updated account with the new primary user, null if account doesnot exist or if user does not exist

UpdateUsername(int, string)

updates the username of the user with `userId` to `username`

```
User? UpdateUsername(int userId, string username)
```

Parameters

`userId` int

`username` string

Returns

User

the updated user with the new username, null if the user does not exist

Namespace BankBackend.Service

Classes

AccountIdNotFoundException

BankExceptions

BankService

InsufficientFundsException

InvalidPasswordException

RepositoryException

UserIdNotFoundException

UserNotAuthorizedException

UsernameAlreadyExistsException

UsernameNotFoundException

Interfaces

IBankService

Class AccountIdNotFoundException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class AccountIdNotFoundException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← AccountIdNotFoundException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

AccountIdNotFoundException()

```
public AccountIdNotFoundException()
```

AccountIdNotFoundException(string)

```
public AccountIdNotFoundException(string message)
```

Parameters

`message` string

Class BankExceptions

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class BankExceptions : Exception, ISerializable
```

Inheritance

object ← Exception ← BankExceptions

Implements

ISerializable

Derived

AccountIdNotFoundException, InsufficientFundsException, InvalidPasswordException, RepositoryException, UserIdNotFoundException, UserNotAuthorizedException, UsernameAlreadyExistsException, UsernameNotFoundException

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

BankExceptions()

```
public BankExceptions()
```

BankExceptions(string)

```
public BankExceptions(string message)
```

Parameters

`message` string

Class BankService

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class BankService : IBankService
```

Inheritance

object ← BankService

Implements

IBankService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

BankService(IBankRepository)

constructor for dependency injection

```
public BankService(IBankRepository repository)
```

Parameters

`repository` IBankRepository

Methods

AddAccountUser(int, int, int)

```
public User AddAccountUser(int userId, int addedUser, int accountId)
```

Parameters

`userId` int

`addedUser` int

`accountId` int

Returns

User

CreateAccount(Account)

```
public Account CreateAccount(Account account)
```

Parameters

`account` Account

Returns

Account

CreateUser(User)

creates a user and returns the created user if the given user has any an `UserId` other than 0 the `UserId` is ignored

```
public User CreateUser(User user)
```

Parameters

`user` User

Returns

User

the created user

Deposit(int, int, double)

```
public Transaction Deposit(int userId, int accountId, double amount)
```

Parameters

`userId` int

`accountId` int

`amount` double

Returns

Transaction

GetAccountByAccountId(int)

```
public Account GetAccountByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

Account

GetAccountsByUserId(int)

find

```
public List<Account> GetAccountsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Account>

Exceptions

UserIdNotFoundException

GetAllAccounts()

```
public List<Account> GetAllAccounts()
```

Returns

List<Account>

GetAllUsers()

finds all users

```
public List<User> GetAllUsers()
```

Returns

List<User>

all users

GetTransactionsByAccountId(int)

```
public List<Transaction> GetTransactionsByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

List<Transaction>

GetTransactionsByUserId(int)

```
public List<Transaction> GetTransactionsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Transaction>

GetUserByUserId(int)

finds a user by `userId`

```
public User GetUserByUserId(int userId)
```

Parameters

`userId` int

Returns

User

Exceptions

UserIdNotFoundException

if a user with the userId does not exist

GetUserByUsername(string)

finds a user by `username`

```
public User GetUserByUsername(string username)
```

Parameters

`username` string

Returns

User

user with the username

Exceptions

UsernameNotFoundException

if a user with the username does not exist

RemoveAccountUser(int, int)

```
public User RemoveAccountUser(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

User

Transfer(int, int, int, double)

```
public Transaction Transfer(int userId, int fromAccountId, int toAccountId,
double amount)
```

Parameters

`userId` int

`fromAccountId` int

`toAccountId` int

`amount` double

Returns

Transaction

UpdateUserProfile(int, string, string)

```
public User UpdateUserProfile(int userId, string newUsername, string newPassword)
```

Parameters

`userId` int

`newUsername` string

`newPassword` string

Returns

User

ValidateLogin(string, string)

validates user loging with `username` and`password`

```
public User ValidateLogin(string username, string password)
```

Parameters

`username` string

`password` string

Returns

User

the logged in user if credentials are correct

Exceptions

UsernameNotFoundException

if user name does not exist

InvalidPasswordException

if password is incorrect

Withdraw(int, int, double)

```
public Transaction Withdraw(int userId, int accountId, double amount)
```

Parameters

`userId` int

`accountId` int

`amount` double

Returns

Transaction

Interface IBankService

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public interface IBankService
```

Methods

AddAccountUser(int, int, int)

```
User AddAccountUser(int userId, int addedUser, int accountId)
```

Parameters

`userId` int

`addedUser` int

`accountId` int

Returns

User

CreateAccount(Account)

```
Account CreateAccount(Account account)
```

Parameters

`account` Account

Returns

Account

CreateUser(User)

```
User CreateUser(User user)
```

Parameters

`user` User

Returns

User

Deposit(int, int, double)

```
Transaction Deposit(int userId, int accountId, double amount)
```

Parameters

`userId` int

`accountId` int

`amount` double

Returns

Transaction

GetAccountByAccountId(int)

```
Account GetAccountByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

Account

GetAccountsByUserId(int)

```
List<Account> GetAccountsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Account>

GetAllAccounts()

```
List<Account> GetAllAccounts()
```

Returns

List<Account>

GetAllUsers()

```
List<User> GetAllUsers()
```

Returns

List<User>

GetTransactionsByAccountId(int)

```
List<Transaction> GetTransactionsByAccountId(int accountId)
```

Parameters

`accountId` int

Returns

List<Transaction>

GetTransactionsByUserId(int)

```
List<Transaction> GetTransactionsByUserId(int userId)
```

Parameters

`userId` int

Returns

List<Transaction>

GetUserByUserId(int)

```
User GetUserByUserId(int userId)
```

Parameters

`userId` int

Returns

User

GetUserByUsername(string)

```
User GetUserByUsername(string username)
```

Parameters

`username` string

Returns

User

RemoveAccountUser(int, int)

```
User RemoveAccountUser(int userId, int accountId)
```

Parameters

`userId` int

`accountId` int

Returns

User

Transfer(int, int, int, double)

```
Transaction Transfer(int userId, int fromAccountId, int toAccountId, double amount)
```

Parameters

`userId` int

`fromAccountId` int

`toAccountId` int

`amount` double

Returns

Transaction

UpdateUserProfile(int, string, string)

```
User UpdateUserProfile(int userId, string newUsername, string newPassword)
```

Parameters

`userId` int

`newUsername` string

`newPassword` string

Returns

User

ValidateLogin(string, string)

```
User ValidateLogin(string username, string password)
```

Parameters

`username` string

`password` string

Returns

User

Withdraw(int, int, double)

```
Transaction Withdraw(int userId, int accountId, double amount)
```

Parameters

`userId` int

`accountId` int

`amount` double

Returns

Transaction

Class InsufficientFundsException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class InsufficientFundsException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← InsufficientFundsException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

InsufficientFundsException()

```
public InsufficientFundsException()
```

InsufficientFundsException(string)

```
public InsufficientFundsException(string message)
```

Parameters

`message` string

Class InvalidPasswordException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class InvalidPasswordException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← InvalidPasswordException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

InvalidPasswordException()

```
public InvalidPasswordException()
```

InvalidPasswordException(string)

```
public InvalidPasswordException(string message)
```

Parameters

`message` string

Class RepositoryException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class RepositoryException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← RepositoryException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

RepositoryException()

```
public RepositoryException()
```

RepositoryException(string)

```
public RepositoryException(string message)
```

Parameters

`message` string

Class UserIdNotFoundException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class UserIdNotFoundException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← UserIdNotFoundException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

UserIdNotFoundException()

```
public UserIdNotFoundException()
```

UserIdNotFoundException(string)

```
public UserIdNotFoundException(string message)
```

Parameters

`message` string

Class UserNotAuthorizedException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class UserNotAuthorizedException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← UserNotAuthorizedException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

UserNotAuthorizedException()

```
public UserNotAuthorizedException()
```

UserNotAuthorizedException(string)

```
public UserNotAuthorizedException(string message)
```

Parameters

`message` string

Class UsernameAlreadyExistsException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class UsernameAlreadyExistsException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← UsernameAlreadyExistsException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

UsernameAlreadyExistsException()

```
public UsernameAlreadyExistsException()
```

UsernameAlreadyExistsException(string)

```
public UsernameAlreadyExistsException(string message)
```

Parameters

`message` string

Class UsernameNotFoundException

Namespace: BankBackend.Service

Assembly: BankBackend.dll

```
public class UsernameNotFoundException : BankExceptions, ISerializable
```

Inheritance

object ← Exception ← BankExceptions ← UsernameNotFoundException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

UsernameNotFoundException()

```
public UsernameNotFoundException()
```

UsernameNotFoundException(string)

```
public UsernameNotFoundException(string message)
```

Parameters

`message` string