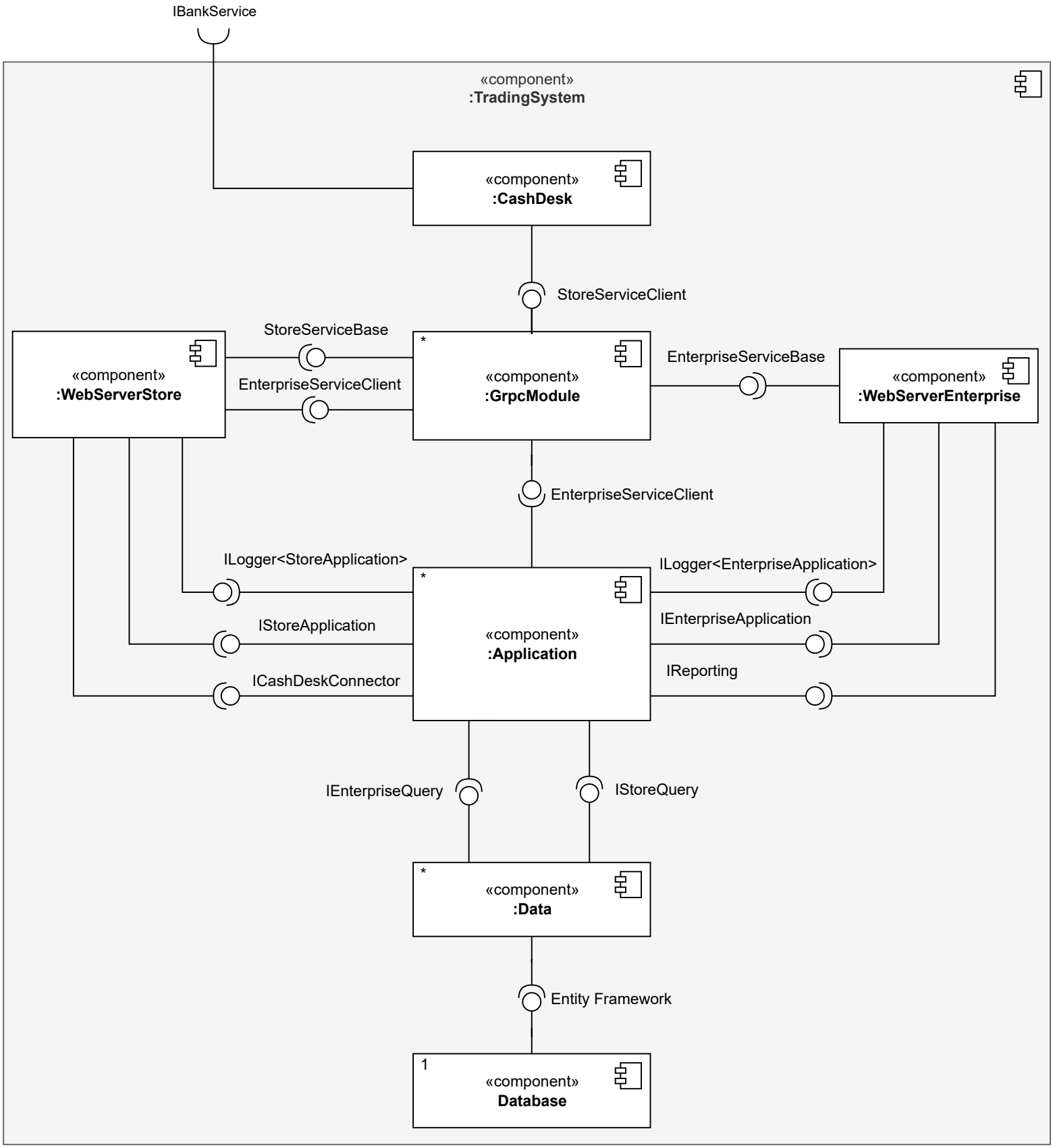
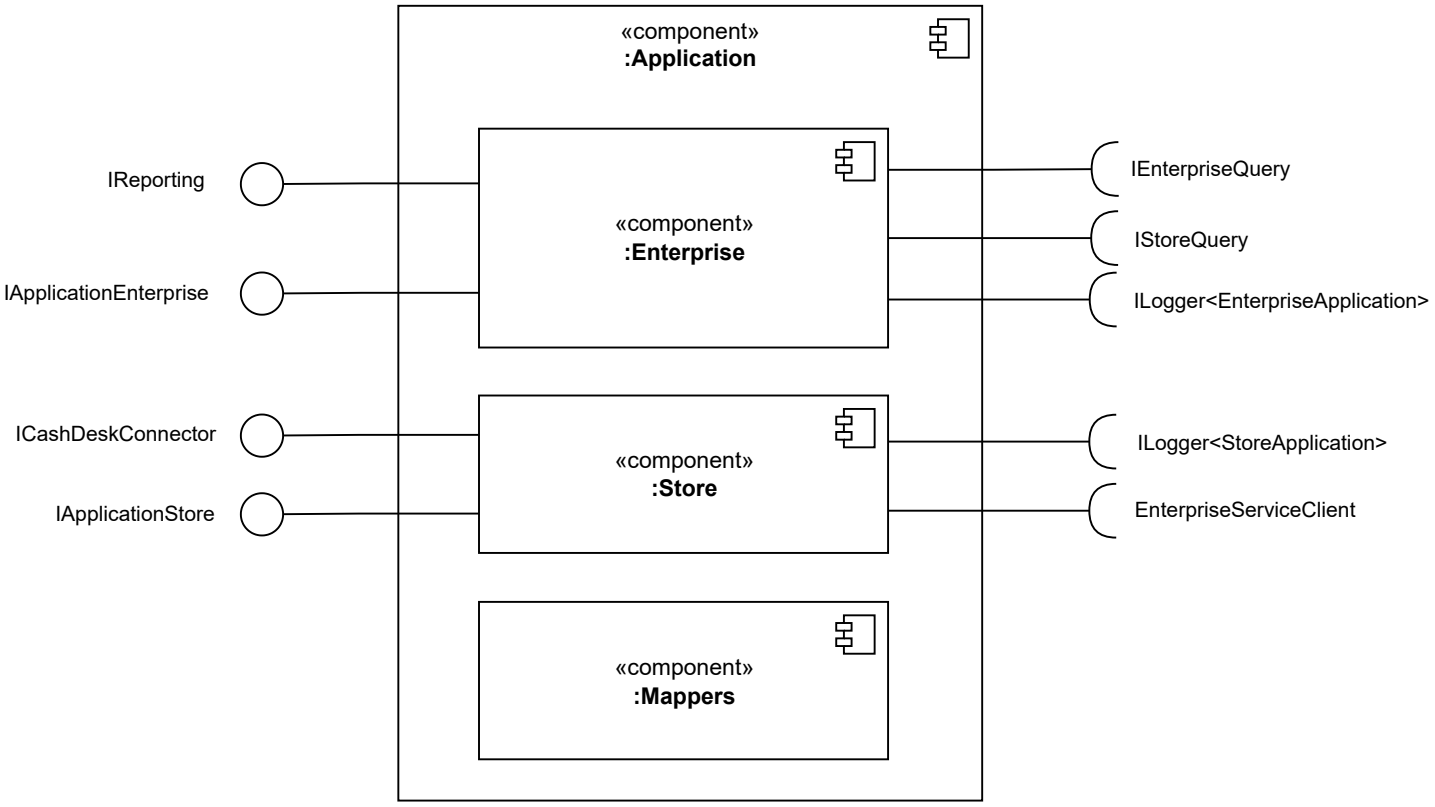
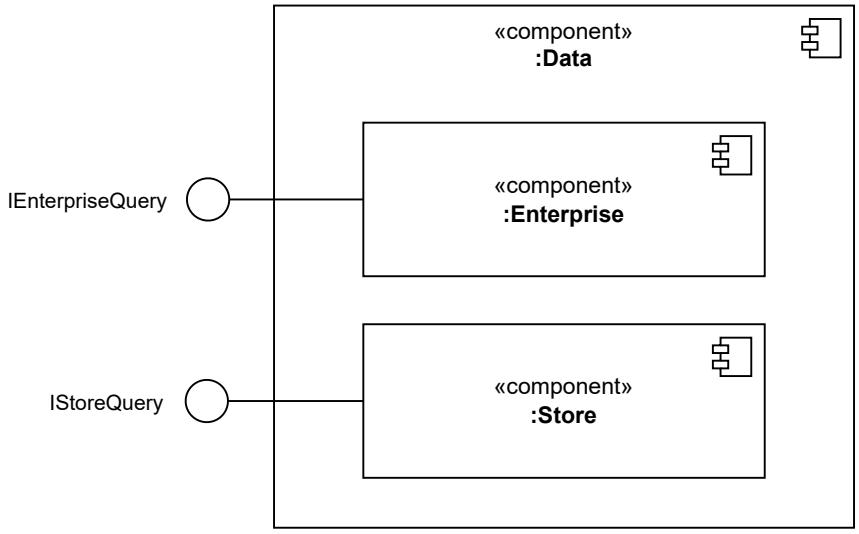
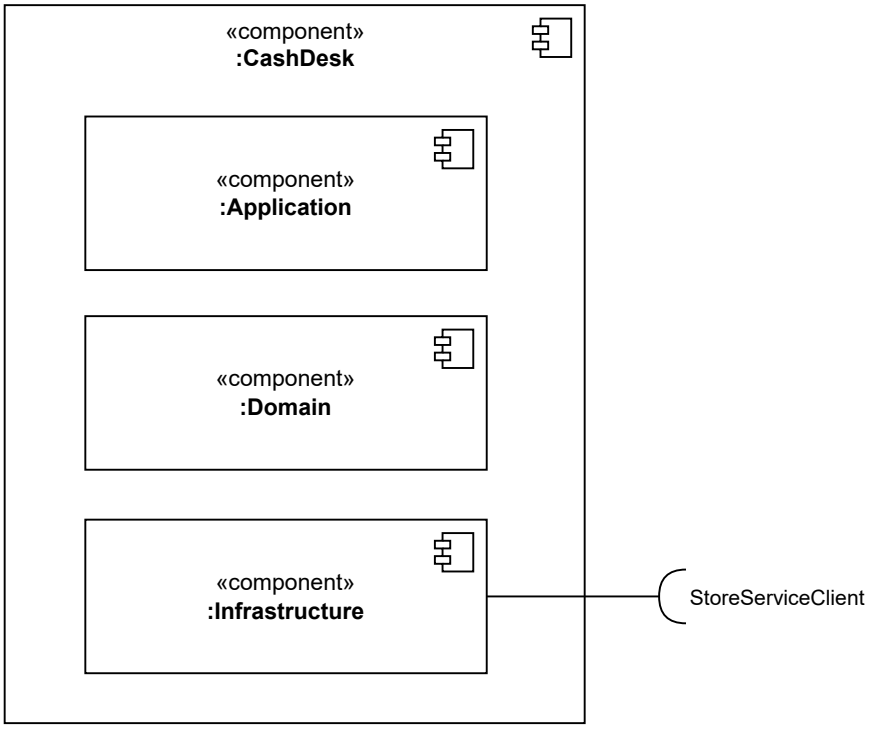
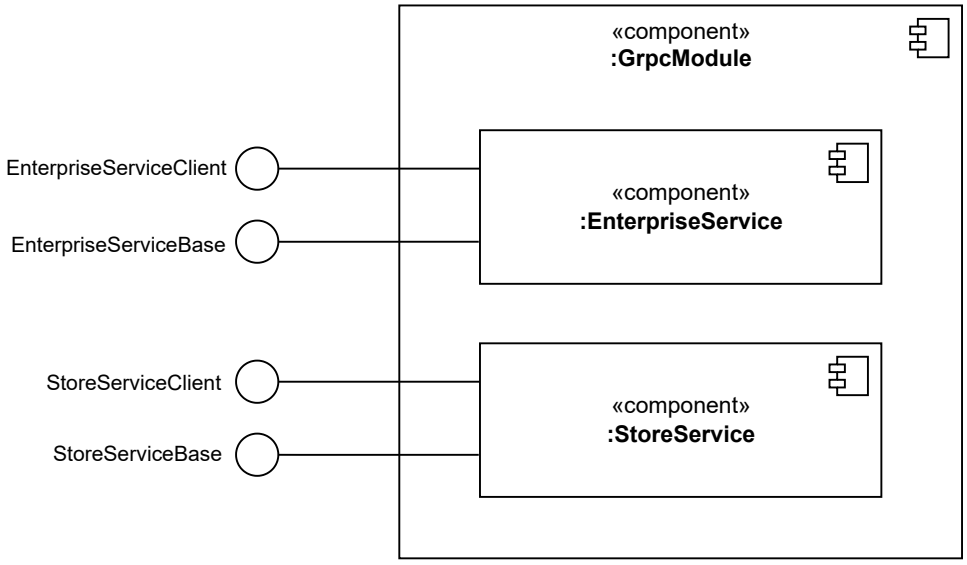
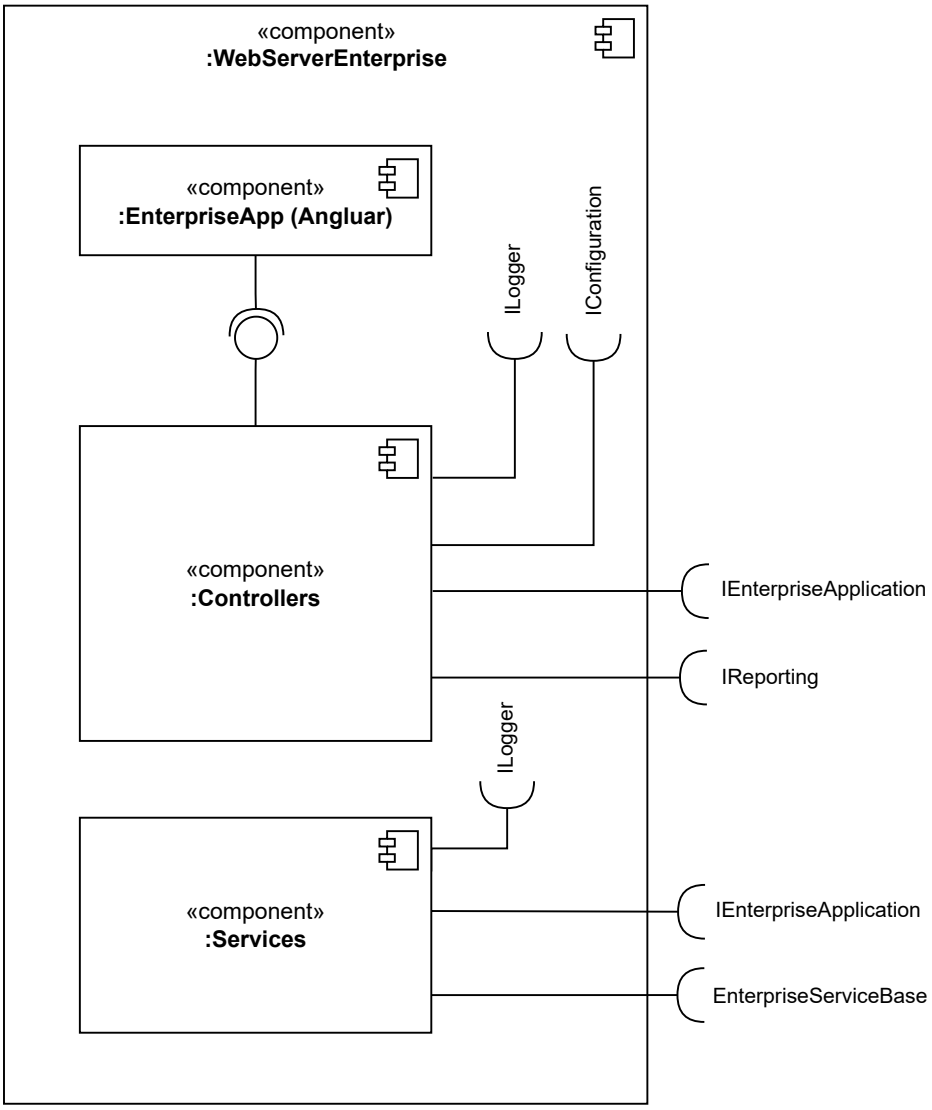

IBankService
«component»
:TradingSystem
«component»
:CashDesk
StoreServiceClient
StoreServiceBase
EnterpriseServiceClient
«component»
:WebServerStore
*
«component»
:GrpcModule
EnterpriseServiceBase
«component»
:WebServerEnterprise
EnterpriseServiceClient
ILogger<StoreApplication>
ILogger<EnterpriseApplication>
IStoreApplication
IEnterpriseApplication
«component»
:Application
ICashDeskConnector
IReporting
IEnterpriseQuery
IStoreQuery
*
«component»
:Data
Entity Framework
1
«component»
Database

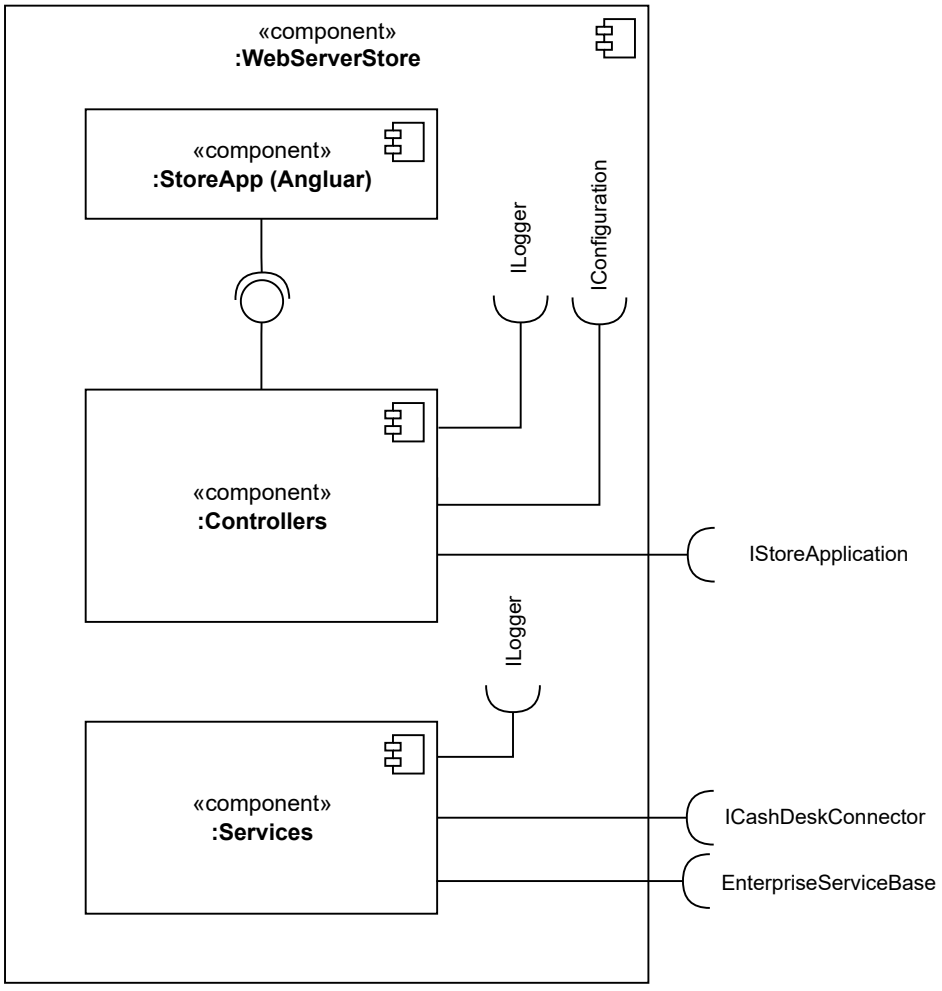

«component»
:Application
IReporting
IApplicationEnterprise
«component»
:Enterprise
IEnterpriseQuery
IStoreQuery
ILogger<EnterpriseApplication>
ICashDeskConnector
IApplicationStore
«component»
:Store
ILogger<StoreApplication>
EnterpriseServiceClient
«component»
:Mappers

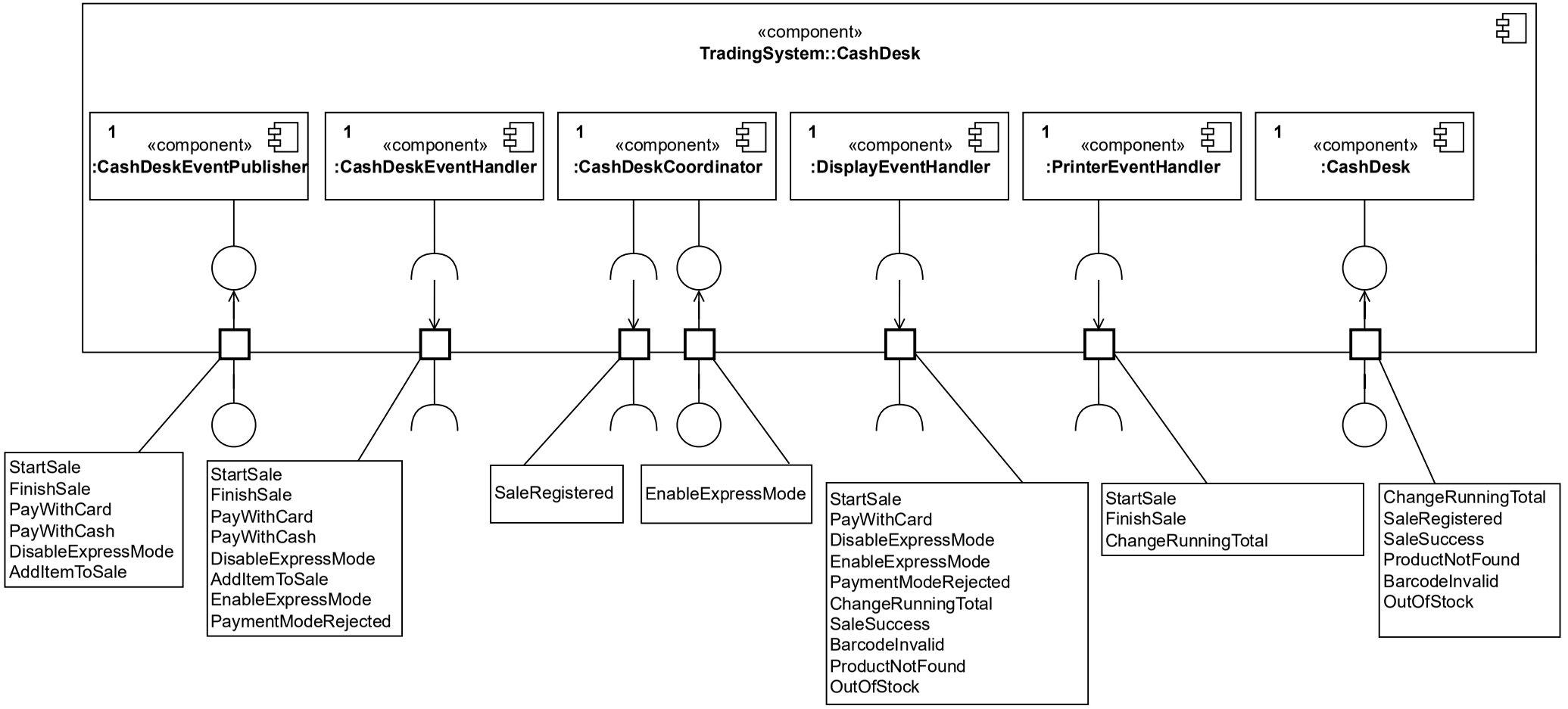

«component»
:Data
«component»
:Enterprise
«component»
:Store
IEnterpriseQuery
IStoreQuery

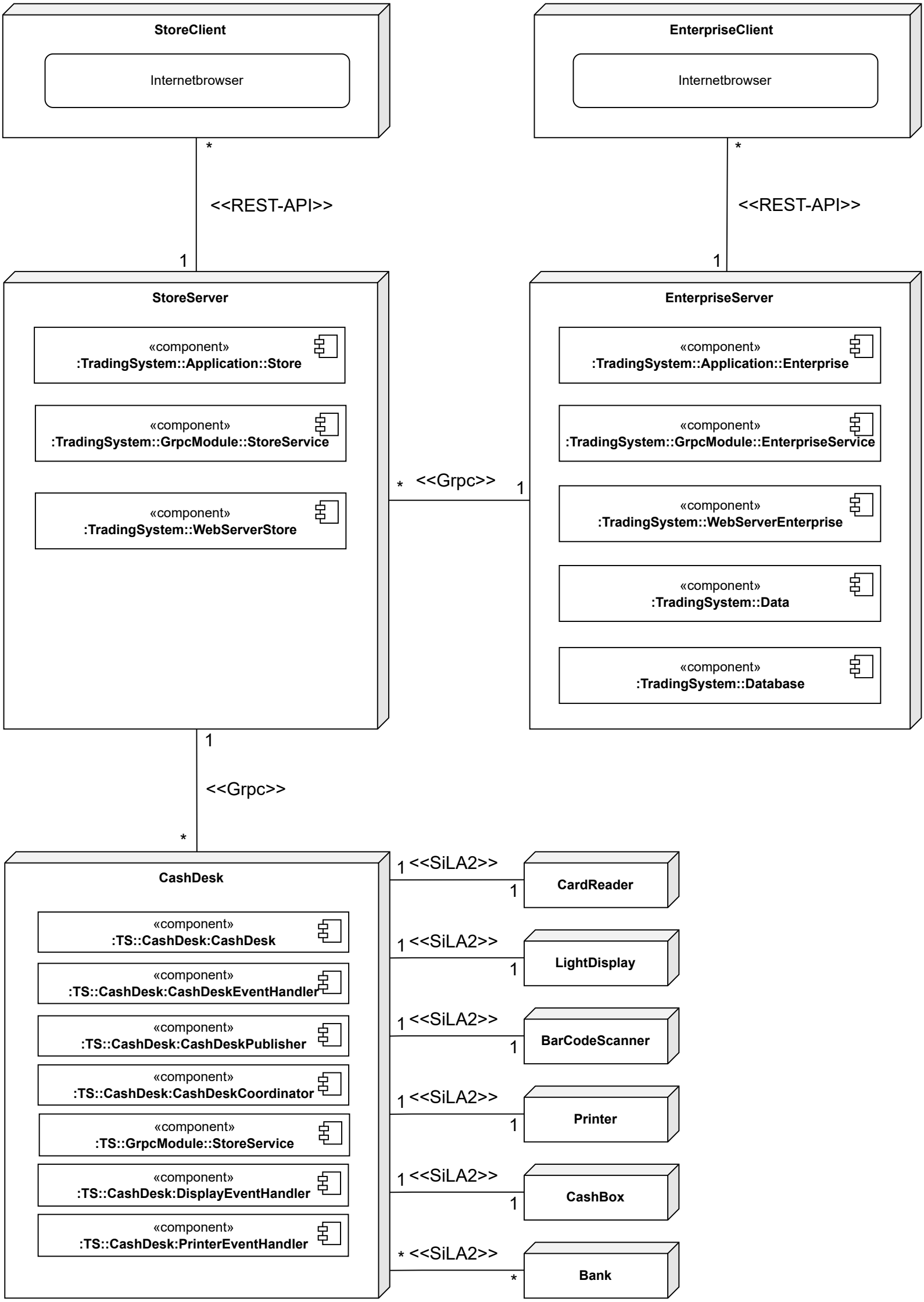

«component»
:CashDesk
«component»
:Application
«component»
:Domain
«component»
:Infrastructure
StoreServiceClient

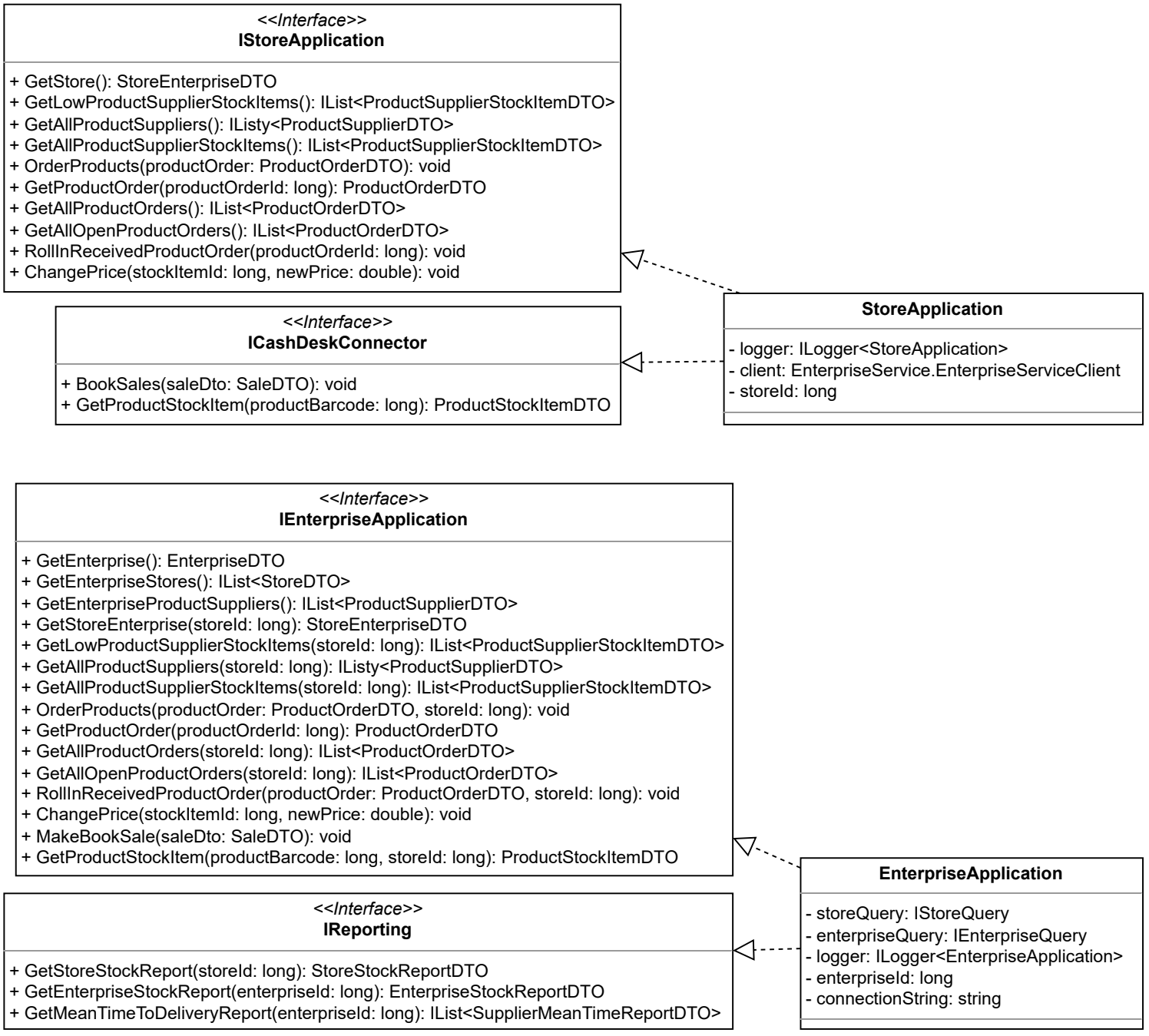

«component»
:GrpcModule
«component»
:EnterpriseService
«component»
:StoreService
EnterpriseServiceClient
EnterpriseServiceBase
StoreServiceClient
StoreServiceBase

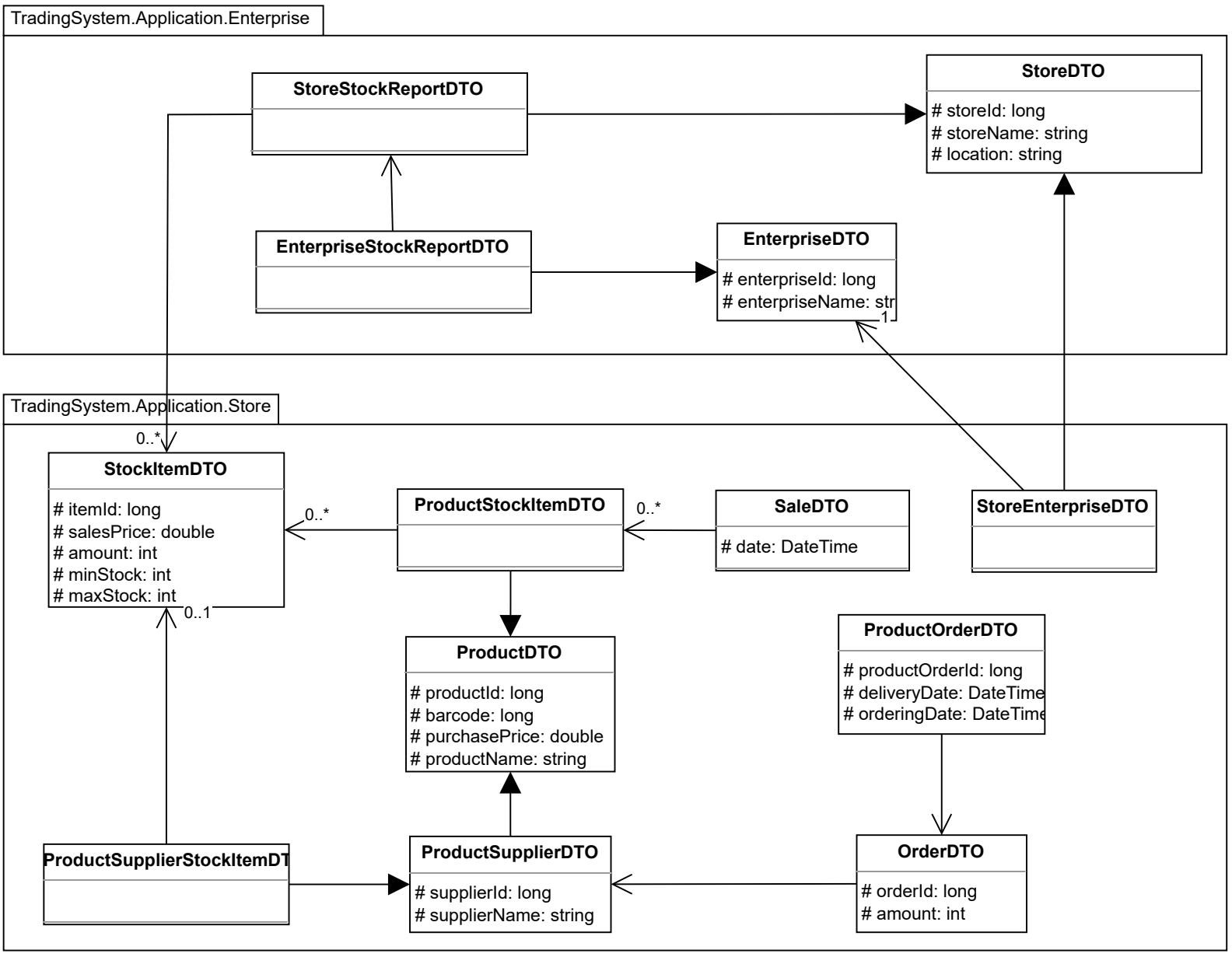

«component»
:WebServerEnterprise
«component»
:EnterpriseApp (Angluar)
ILogger
IConfiguration
«component»
:Controllers
IEnterpriseApplication
IReporting
ILogger
«component»
:Services
IEnterpriseApplication
EnterpriseServiceBase

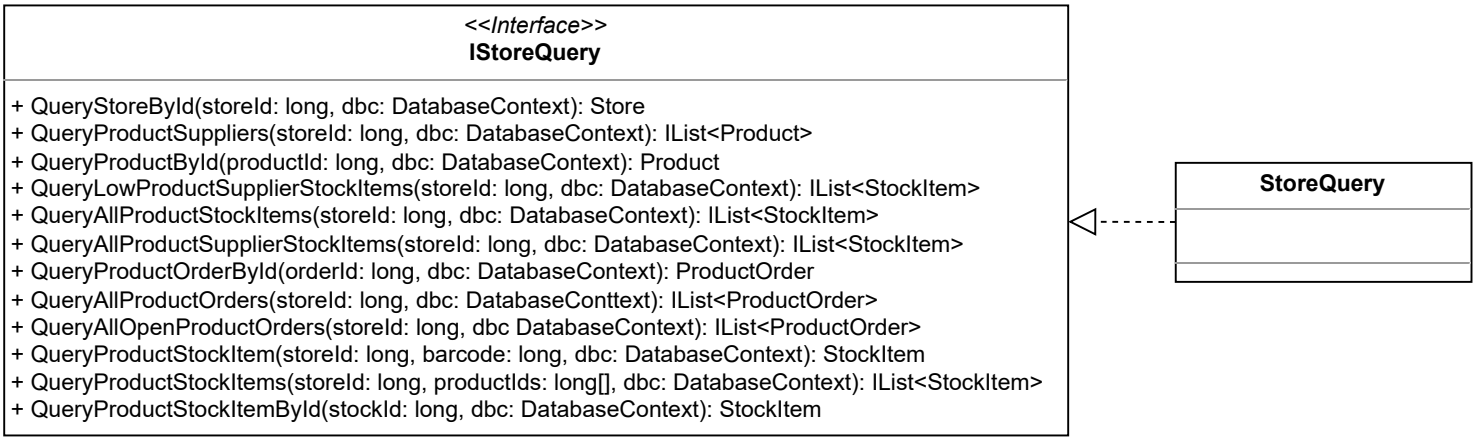

«component»
:WebServerStore
«component»
:StoreApp (Angluar)
ILogger
IConfiguration
«component»
:Controllers
IStoreApplication
ILogger
«component»
:Services
ICashDeskConnector
EnterpriseServiceBase

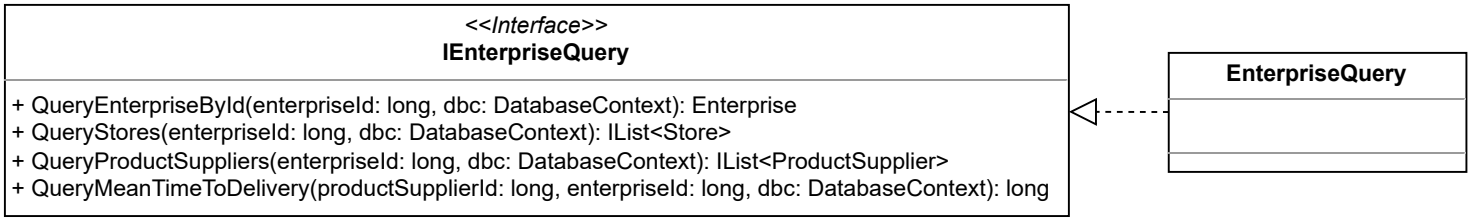

«component»
TradingSystem::CashDesk
1
«component»
:CashDeskEventPublisher
1
«component»
:CashDeskEventHandler
1
«component»
:CashDeskCoordinator
1
«component»
:DisplayEventHandler
1
«component»
:PrinterEventHandler
1
«component»
:CashDesk
StartSale
FinishSale
PayWithCard
PayWithCash
DisableExpressMode
AddItemToSale
StartSale
FinishSale
PayWithCard
PayWithCash
DisableExpressMode
AddItemToSale
EnableExpressMode
PaymentModeRejected
SaleRegistered
EnableExpressMode
StartSale
PayWithCard
DisableExpressMode
EnableExpressMode
PaymentModeRejected
ChangeRunningTotal
SaleSuccess
BarcodeInvalid
ProductNotFound
OutOfStock
StartSale
FinishSale
ChangeRunningTotal
ChangeRunningTotal
SaleRegistered
SaleSuccess
ProductNotFound
BarcodeInvalid
OutOfStock

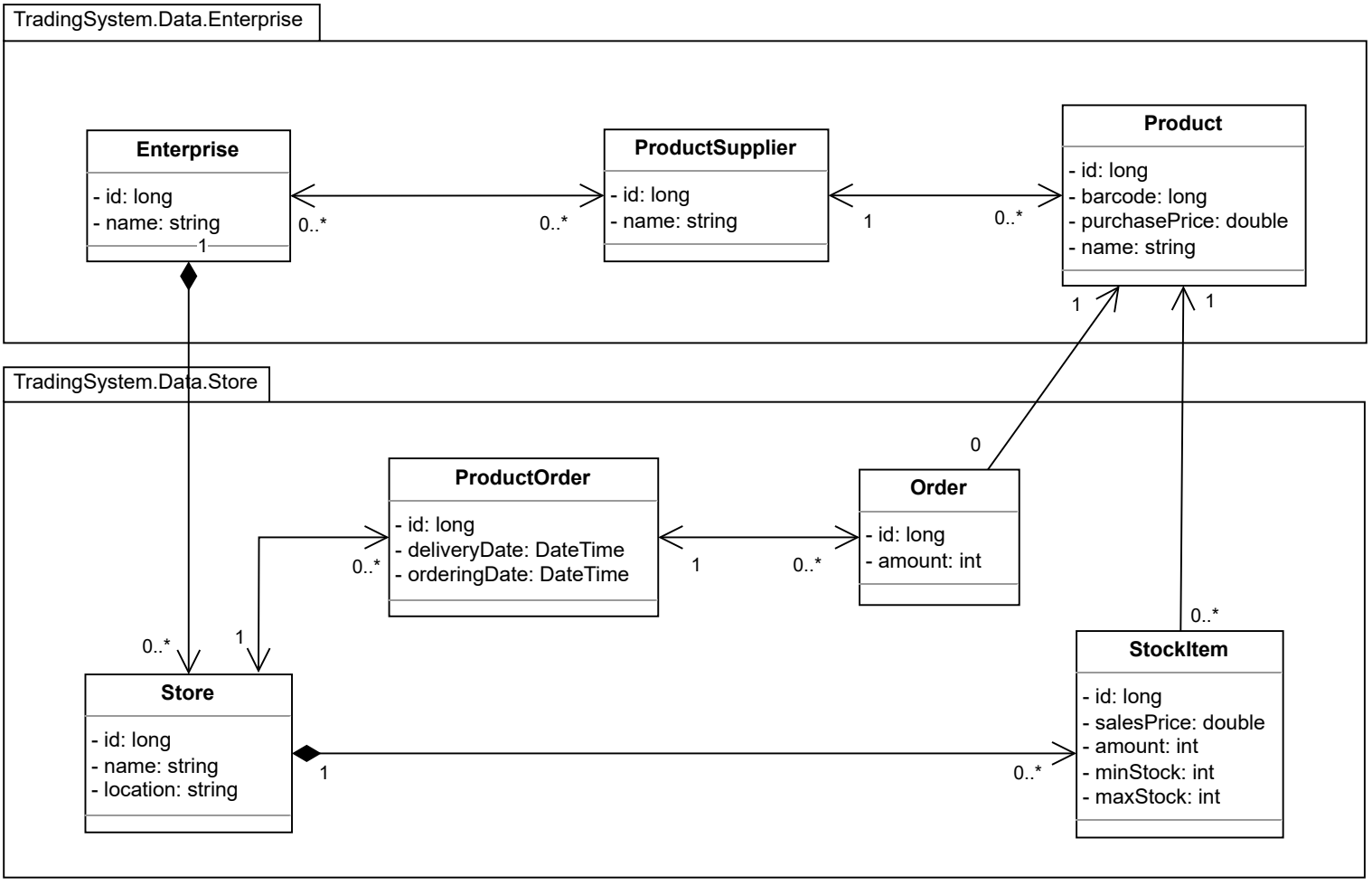

StoreClient
Internetbrowser
EnterpriseClient
Internetbrowser
*
<<REST-API>>
1
*
<<REST-API>>
1
StoreServer
«component»
:TradingSystem::Application::Store
«component»
:TradingSystem::GrpcModule::StoreService
«component»
:TradingSystem::WebServerStore
EnterpriseServer
«component»
:TradingSystem::Application::Enterprise
«component»
:TradingSystem::GrpcModule::EnterpriseService
«component»
:TradingSystem::WebServerEnterprise
«component»
:TradingSystem::Data
«component»
:TradingSystem::Database
* <<Grpc>> 1
1
<<Grpc>>
*
CashDesk
«component»
:TS::CashDesk:CashDesk
«component»
:TS::CashDesk:CashDeskEventHandler
«component»
:TS::CashDesk:CashDeskPublisher
«component»
:TS::CashDesk:CashDeskCoordinator
«component»
:TS::GrpcModule::StoreService
«component»
:TS::CashDesk:DisplayEventHandler
«component»
:TS::CashDesk:PrinterEventHandler
1 <<SiLA2>> 1
CardReader
1 <<SiLA2>> 1
LightDisplay
1 <<SiLA2>> 1
BarCodeScanner
1 <<SiLA2>> 1
Printer
1 <<SiLA2>> 1
CashBox
* <<SiLA2>> *
Bank

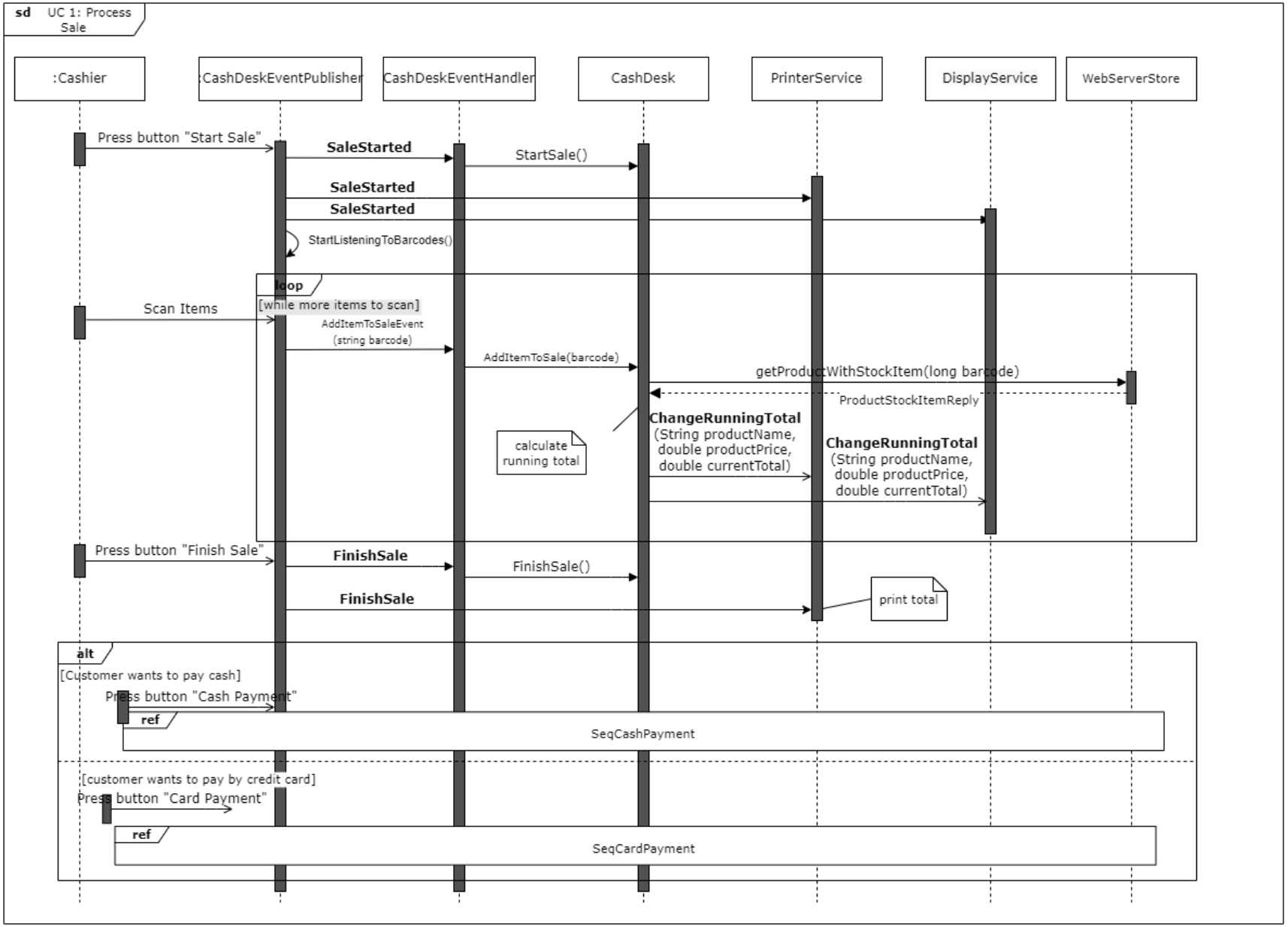

<<Interface>>
IStoreApplication
+ GetStore(): StoreEnterpriseDTO
+ GetLowProductSupplierStockItems(): IList<ProductSupplierStockItemDTO>
+ GetAllProductSuppliers(): IListy<ProductSupplierDTO>
+ GetAllProductSupplierStockItems(): IList<ProductSupplierStockItemDTO>
+ OrderProducts(productOrder: ProductOrderDTO): void
+ GetProductOrder(productOrderId: long): ProductOrderDTO
+ GetAllProductOrders(): IList<ProductOrderDTO>
+ GetAllOpenProductOrders(): IList<ProductOrderDTO>
+ RollInReceivedProductOrder(productOrderId: long): void
+ ChangePrice(stockItemId: long, newPrice: double): void
<<Interface>>
ICashDeskConnector
+ BookSales(saleDto: SaleDTO): void
+ GetProductStockItem(productBarcode: long): ProductStockItemDTO
StoreApplication
- logger: ILogger<StoreApplication>
- client: EnterpriseService.EnterpriseServiceClient
- storeId: long
<<Interface>>
IEnterpriseApplication
+ GetEnterprise(): EnterpriseDTO
+ GetEnterpriseStores(): IList<StoreDTO>
+ GetEnterpriseProductSuppliers(): IList<ProductSupplierDTO>
+ GetStoreEnterprise(storeId: long): StoreEnterpriseDTO
+ GetLowProductSupplierStockItems(storeId: long): IList<ProductSupplierStockItemDTO>
+ GetAllProductSuppliers(storeId: long): IListy<ProductSupplierDTO>
+ GetAllProductSupplierStockItems(storeId: long): IList<ProductSupplierStockItemDTO>
+ OrderProducts(productOrder: ProductOrderDTO, storeId: long): void
+ GetProductOrder(productOrderId: long): ProductOrderDTO
+ GetAllProductOrders(storeId: long): IList<ProductOrderDTO>
+ GetAllOpenProductOrders(storeId: long): IList<ProductOrderDTO>
+ RollInReceivedProductOrder(productOrder: ProductOrderDTO, storeId: long): void
+ ChangePrice(stockItemId: long, newPrice: double): void
+ MakeBookSale(saleDto: SaleDTO): void
+ GetProductStockItem(productBarcode: long, storeId: long): ProductStockItemDTO
<<Interface>>
IReporting
+ GetStoreStockReport(storeId: long): StoreStockReportDTO
+ GetEnterpriseStockReport(enterpriseId: long): EnterpriseStockReportDTO
+ GetMeanTimeToDeliveryReport(enterpriseId: long): IList<SupplierMeanTimeReportDTO>
EnterpriseApplication
- storeQuery: IStoreQuery
- enterpriseQuery: IEnterpriseQuery
- logger: ILogger<EnterpriseApplication>
- enterpriseId: long
- connectionString: string

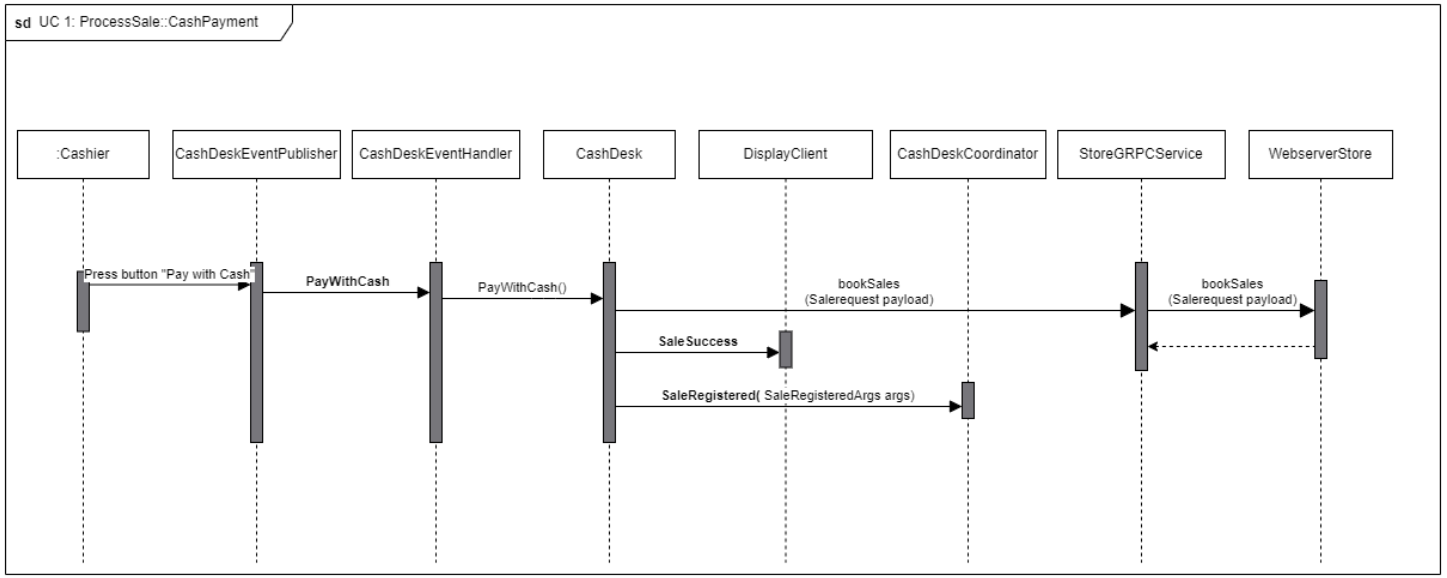

TradingSystem.Application.Enterprise
StoreStockReportDTO
StoreDTO
storeId: long
storeName: string
location: string
EnterpriseStockReportDTO
EnterpriseDTO
enterpriseId: long
enterpriseName: str
1
TradingSystem.Application.Store
0..*
StockItemDTO
itemId: long
salesPrice: double
amount: int
minStock: int
maxStock: int
0..1
0..*
ProductStockItemDTO
0..*
SaleDTO
date: DateTime
StoreEnterpriseDTO
ProductDTO
productId: long
barcode: long
purchasePrice: double
productName: string
ProductOrderDTO
productOrderId: long
deliveryDate: DateTime
orderingDate: DateTime
ProductSupplierStockItemDT
ProductSupplierDTO
supplierId: long
supplierName: string
OrderDTO
orderId: long
amount: int

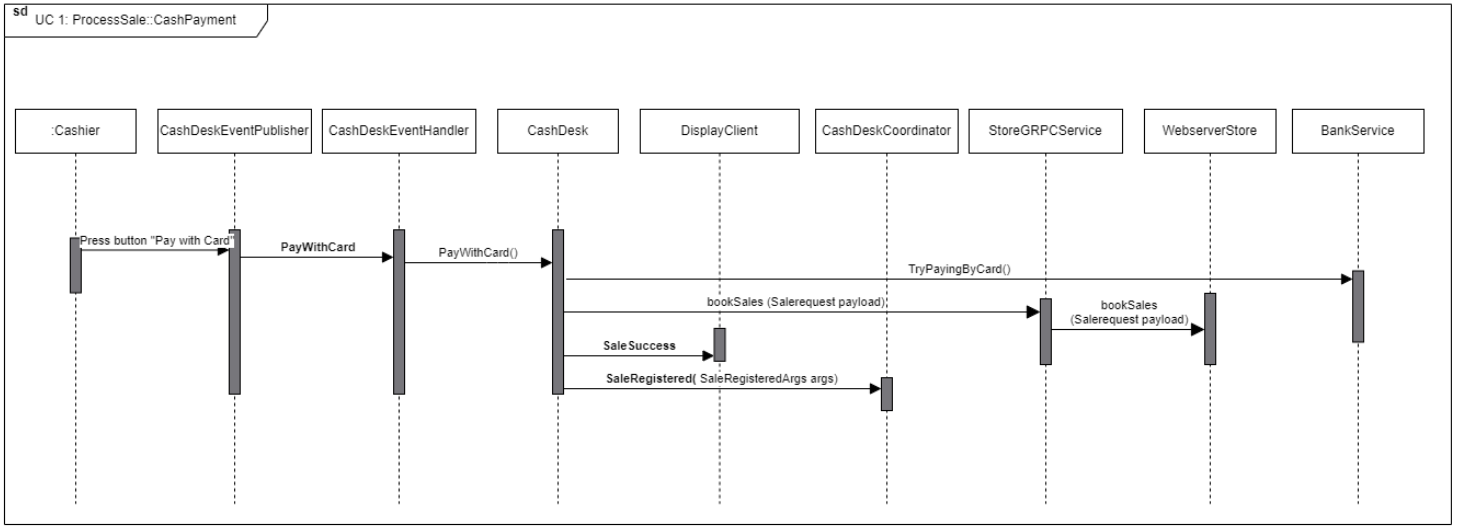

<<Interface>>
IStoreQuery
+ QueryStoreById(storeId: long, dbc: DatabaseContext): Store
+ QueryProductSuppliers(storeId: long, dbc: DatabaseContext): IList<Product>
+ QueryProductById(productId: long, dbc: DatabaseContext): Product
+ QueryLowProductSupplierStockItems(storeId: long, dbc: DatabaseContext): IList<StockItem>
+ QueryAllProductStockItems(storeId: long, dbc: DatabaseContext): IList<StockItem>
+ QueryAllProductSupplierStockItems(storeId: long, dbc: DatabaseContext): IList<StockItem>
+ QueryProductOrderById(orderId: long, dbc: DatabaseContext): ProductOrder
+ QueryAllProductOrders(storeId: long, dbc: DatabaseConttext): IList<ProductOrder>
+ QueryAllOpenProductOrders(storeId: long, dbc DatabaseContext): IList<ProductOrder>
+ QueryProductStockItem(storeId: long, barcode: long, dbc: DatabaseContext): StockItem
+ QueryProductStockItems(storeId: long, productIds: long[], dbc: DatabaseContext): IList<StockItem>
+ QueryProductStockItemById(stockId: long, dbc: DatabaseContext): StockItem
StoreQuery

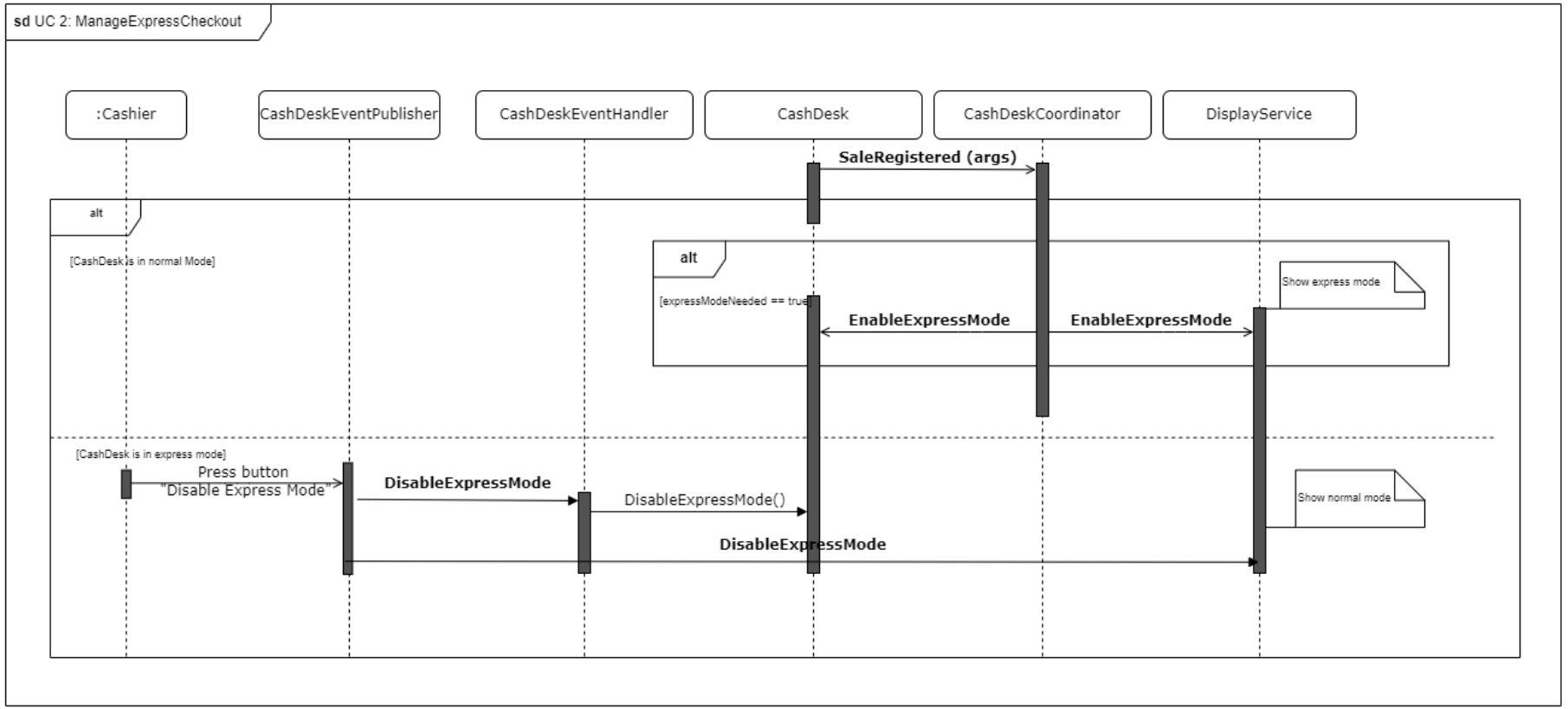

<<Interface>>
IEnterpriseQuery
+ QueryEnterpriseById(enterpriseId: long, dbc: DatabaseContext): Enterprise
+ QueryStores(enterpriseId: long, dbc: DatabaseContext): IList<Store>
+ QueryProductSuppliers(enterpriseId: long, dbc: DatabaseContext): IList<ProductSupplier>
+ QueryMeanTimeToDelivery(productSupplierId: long, enterpriseId: long, dbc: DatabaseContext): long
EnterpriseQuery

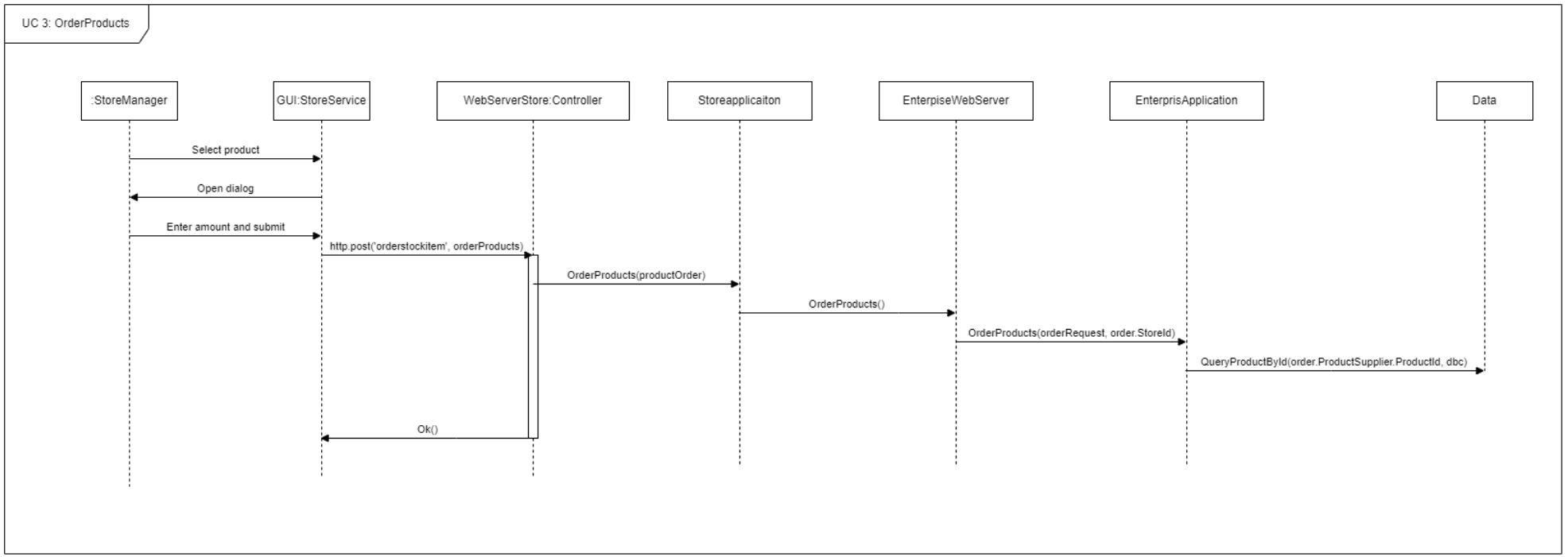

TradingSystem.Data.Enterprise
Enterprise
- id: long
- name: string
ProductSupplier
- id: long
- name: string
Product
- id: long
- barcode: long
- purchasePrice: double
- name: string
0..*
0..*
1
0..*
1
1
1
TradingSystem.Data.Store
ProductOrder
- id: long
- deliveryDate: DateTime
- orderingDate: DateTime
Order
- id: long
- amount: int
0
0..*
1
0..*
0..*
1
0..*
Store
- id: long
- name: string
- location: string
StockItem
- id: long
- salesPrice: double
- amount: int
- minStock: int
- maxStock: int
1
0..*

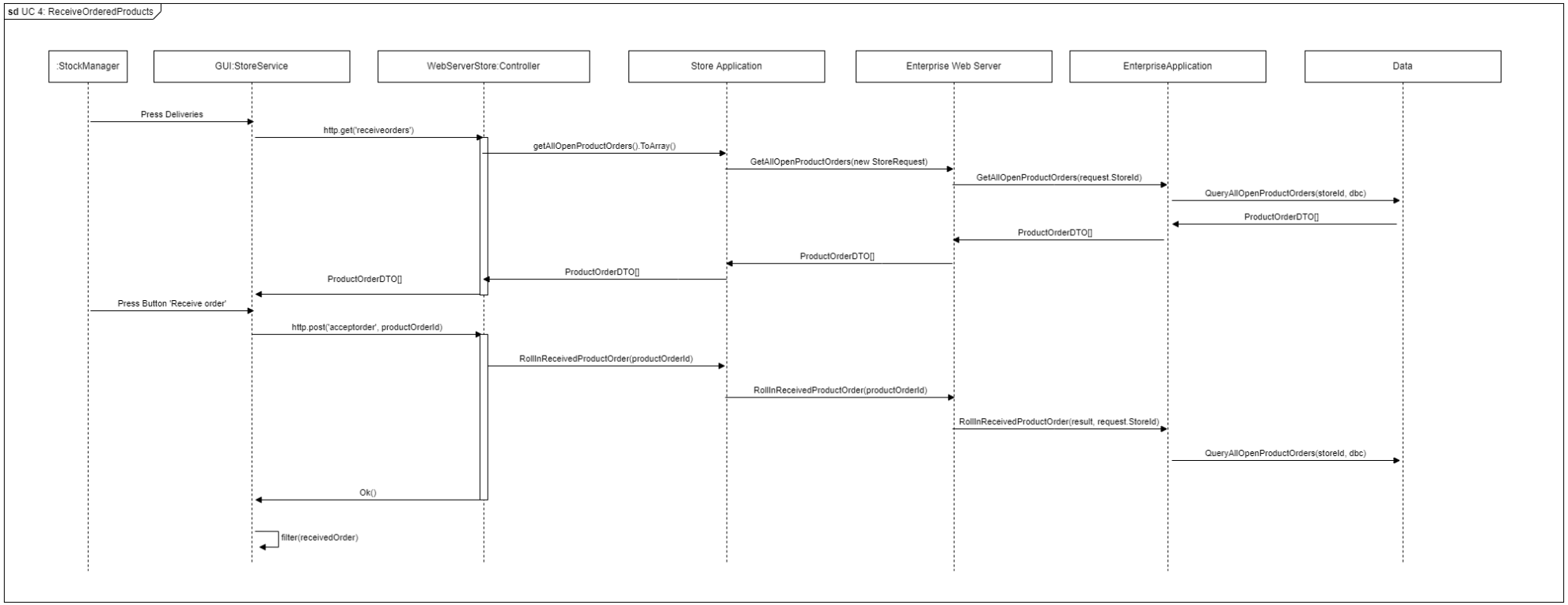

sd UC 1: Process Sale
:Cashier
:CashDeskEventPublisher
CashDeskEventHandler
CashDesk
PrinterService
DisplayService
WebServerStore
Press button "Start Sale"
SaleStarted
StartSale()
SaleStarted
SaleStarted
StartListeningToBarcodes()
loop
[while more items to scan]
Scan Items
AddItemToSaleEvent
(string barcode)
AddItemToSale(barcode)
getProductWithStockItem(long barcode)
ProductStockItemReply
ChangeRunningTotal
(String productName,
double productPrice,
double currentTotal)
calculate
running total
ChangeRunningTotal
(String productName,
double productPrice,
double currentTotal)
Press button "Finish Sale"
FinishSale
FinishSale()
FinishSale
print total
alt
[Customer wants to pay cash]
Press button "Cash Payment"
ref
SeqCashPayment
[customer wants to pay by credit card]
Press button "Card Payment"
ref
SeqCardPayment

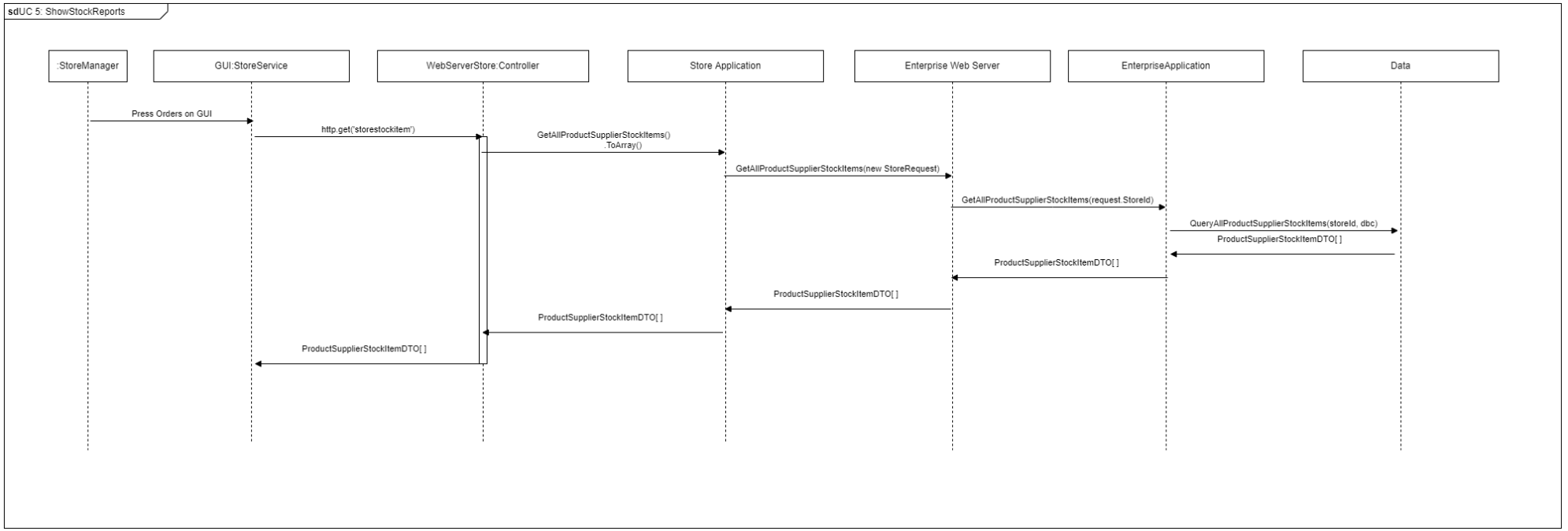

sd UC 1: ProcessSale::CashPayment
:Cashier
CashDeskEventPublisher
CashDeskEventHandler
CashDesk
DisplayClient
CashDeskCoordinator
StoreGRPCService
WebserverStore
Press button "Pay with Cash"
PayWithCash
PayWithCash()
bookSales
(Salerequest payload)
bookSales
(Salerequest payload)
SaleSuccess
SaleRegistered(SaleRegisteredArgs args)

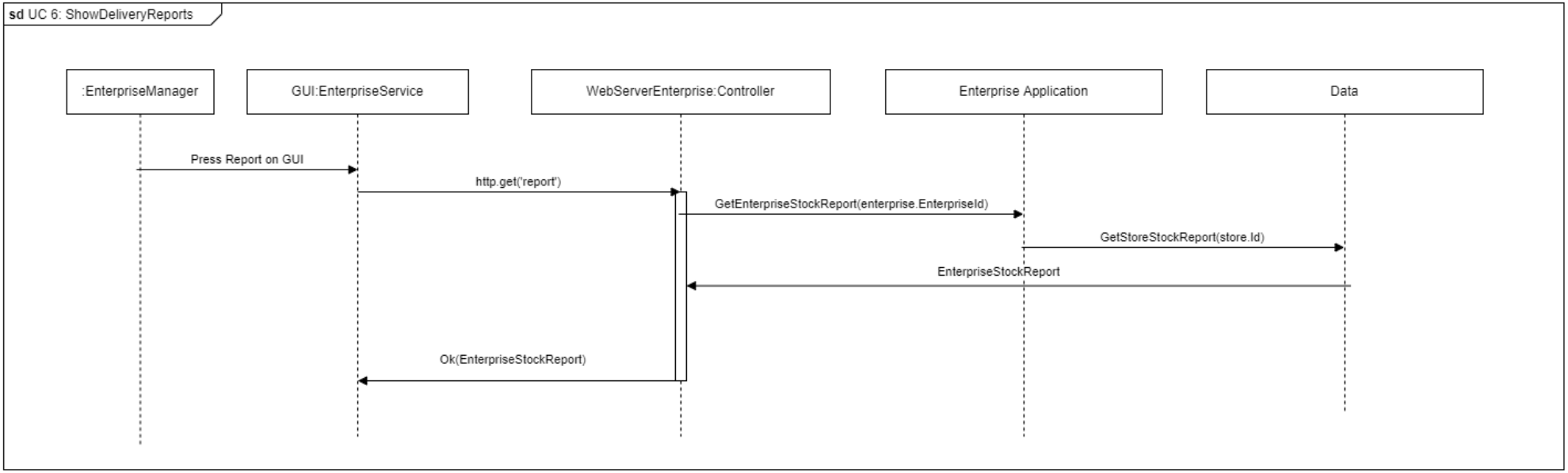

sd UC 1: ProcessSale::CashPayment
:Cashier
CashDeskEventPublisher
CashDeskEventHandler
CashDesk
DisplayClient
CashDeskCoordinator
StoreGRPCService
WebserverStore
BankService
Press button "Pay with Card"
PayWithCard
PayWithCard()
TryPayingByCard()
bookSales (Salerequest payload);
bookSales
(Salerequest payload)
SaleSuccess
SaleRegistered(SaleRegisteredArgs args)

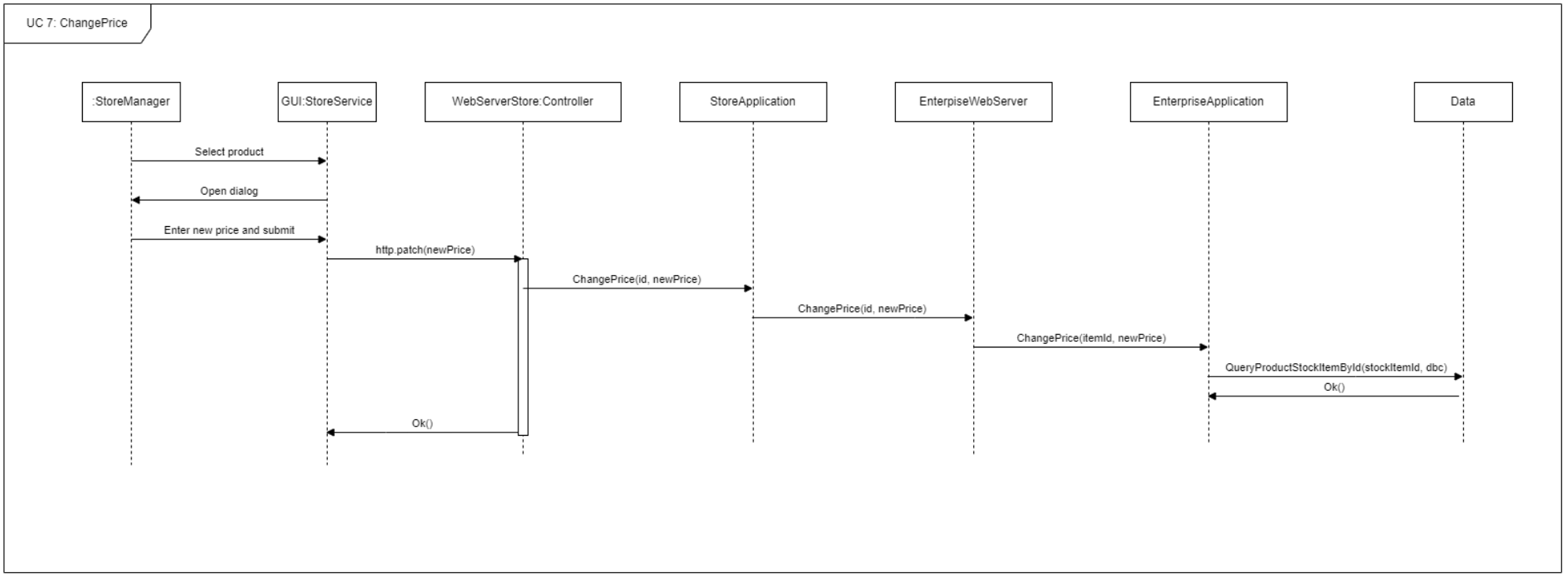

sd UC 2: ManageExpressCheckout
:Cashier
CashDeskEventPublisher
CashDeskEventHandler
CashDesk
CashDeskCoordinator
DisplayService
SaleRegistered (args)
alt
[CashDesk is in normal Mode]
alt
[expressModeNeeded == true]
Show express mode
EnableExpressMode
EnableExpressMode
[CashDesk is in express mode]
Press button
"Disable Express Mode"
DisableExpressMode
DisableExpressMode()
Show normal mode
DisableExpressMode


UC 3: OrderProducts
:StoreManager
GUI:StoreService
WebServerStore:Controller
Storeapplicaiton
EnterpiseWebServer
EnterprisApplication
Data
Select product
Open dialog
Enter amount and submit
http.post('orderstockitem', orderProducts)
OrderProducts(productOrder)
OrderProducts()
OrderProducts(orderRequest, order.StoreId)
QueryProductById(order.ProductSupplier.ProductId, dbc)
Ok()


sd UC 4: ReceiveOrderedProducts
:StockManager
GUI:StoreService
WebServerStore:Controller
Store Application
Enterprise Web Server
EnterpriseApplication
Data
Press Deliveries
http.get('receiveorders')
getAllOpenProductOrders().ToArray()
GetAllOpenProductOrders(new StoreRequest)
GetAllOpenProductOrders(request.StoreId)
QueryAllOpenProductOrders(storeId, dbc)
ProductOrderDTO[]
ProductOrderDTO[]
ProductOrderDTO[]
ProductOrderDTO[]
ProductOrderDTO[]
Press Button 'Receive order'
http.post('acceptorder', productOrderId)
RollInReceivedProductOrder(productOrderId)
RollInReceivedProductOrder(productOrderId)
RollInReceivedProductOrder(result, request.StoreId)
QueryAllOpenProductOrders(storeId, dbc)
Ok()
filter(receivedOrder)


sdUC 5: ShowStockReports
:StoreManager
GUI:StoreService
WebServerStore:Controller
Store Application
Enterprise Web Server
EnterpriseApplication
Data
Press Orders on GUI
http.get('storestockitem')
GetAllProductSupplierStockItems()
.ToArray()
GetAllProductSupplierStockItems(new StoreRequest)
GetAllProductSupplierStockItems(request.StoreId)
QueryAllProductSupplierStockItems(storeId, dbc)
ProductSupplierStockItemDTO[ ]
ProductSupplierStockItemDTO[ ]
ProductSupplierStockItemDTO[ ]
ProductSupplierStockItemDTO[ ]
ProductSupplierStockItemDTO[ ]


sd UC 6: ShowDeliveryReports
:EnterpriseManager
GUI:EnterpriseService
WebServerEnterprise:Controller
Enterprise Application
Data
Press Report on GUI
http.get('report')
GetEnterpriseStockReport(enterprise.EnterpriseId)
GetStoreStockReport(store.Id)
EnterpriseStockReport
Ok(EnterpriseStockReport)


UC 7: ChangePrice
:StoreManager
GUI:StoreService
WebServerStore:Controller
StoreApplication
EnterpiseWebServer
EnterpriseApplication
Data
Select product
Open dialog
Enter new price and submit
http.patch(newPrice)
ChangePrice(id, newPrice)
ChangePrice(id, newPrice)
ChangePrice(itemId, newPrice)
QueryProductStockItemById(stockItemId, dbc)
Ok()
Ok()