

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Цэдэн-Ишийн Биндэрцэцэг

**Программ хангамжийн зохиомжийн үлгэр
загварууд ба түүний хэрэглээ**
(Software design patterns and its application)

Программ Хангамж (D061302)
Үйлдвэрлэлийн дадлагын тайлан

Улаанбаатар хот

2025 оны 9 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

Программ хангамжийн зохиомжийн үлгэр загварууд ба
түүний хэрэглээ
(Software design patterns and its application)

Программ Хангамж (D061302)
Үйлдвэрлэлийн дадлагын тайлан

Удирдагч: _____ Х. Ганзориг
Хамтран удирдагч: _____ Б. Батням
Гүйцэтгэсэн: _____ Ц. Биндэрцэцэг (22B1NUM0027)

Улаанбаатар хот

2025 оны 9 сар

Зохиогчийн баталгаа

Миний бие Цэдэн-Ишийн Биндэрцэцэг нь ”Программ хангамжийн зохиомжийн үлгэр загварууд ба түүний хэрэглээ” сэдэвтэй дадлагын ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Энэхүү ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулаагүй болно.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дадлагын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

ҮЙЛДВЭРЛЭЛИЙН ДАДЛАГА АЖЛЫН ТАЙЛАН

Үйлдвэрлэлийн дадлага хийсэн
байгууллагын нэр:

Шалгасан:

Үйлдвэрлэлийн дадлага удирдсан
ажилтны нэр, албан тушаал:

Гүйцэтгэсэн:
Програм хангамжийн
III ангийн оюутан

Монгол Ай-Ди (Одигитрия
Улаанбаатар хот)

Х. Ганзориг, СТО /.....

Ц. Биндэрцэцэг /.....
Ч. Биндэрцэцэг



Улаанбаатар хот
2025 он

МУИС, МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ

.....Прожил хамгийн.....АНГИЙН

ОЮУТАН ..Бик дэлхүүч.. -ИЙН

ҮЙЛДВЭРЛЭЛИЙН ДАДЛАГЫН АЖЛЫН УДИРДАГЧИЙН ТОДОРХОЙЛОЛТ

2025 оны 09 сарын 01

.....Прожил хамгийнкодтой оюутанч.бийцүүлэх.... нь манай байгууллагад 2025 оны 08 сарын 11-ны өдрөөс 08 сарын 19-ны өдөр хүртэл мэргэшүүлэх дадлагыг батлагдсан удирдамж, ажлын төлөвлөгөөний дагуу гүйцэтгэлээ.

Оюутан-ын удирдамжийн дагуу дадлагын ажлыг гүйцэтгэсэн байдал:

.....Дадлагынхариуцалтуудогултаныөвлийн
.....жилласанажлыгхарчмын талынчалал
.....гүйцэтгэжбиедэсийнажилтадгэгээсүүсээ
.....харчмынмөхөвлийнажилтадажилтад
.....жилласантөлөвлөлийнөргөжсөнхариуцалтууд т.
.....төлөвлөжсөнжилүүдэшигүүрчүүлэхжилласан
.....хэрэгжүүлэх сан.

Үнэлгээний санал: Үйлдвэрлэлийн дадлагын хариуцсан удирдагчийн өгөх оноо

(10 онооноос өгнө)

10.

Үйлдвэрлэлийн дадлагын хариуцсан удирдагч1 / X. Газарж /



МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
ПРОГРАМ ХАНГАМЖИЙН III АНГИЙН ОЮУТАН
Ц. БИНДЭРЦЭЦЭГ

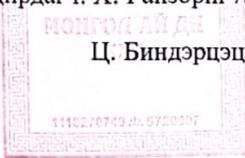
ДАДЛАГЫН АЖЛЫН ТӨЛӨВЛӨГӨӨ

2025 он 09 сар 01 өдөр

№	Гүйцэтгэх ажил	Хугацаа	Биселт	Удирдагчийн үнэлгээ
1	Програм хангамжийн үлгэр загваруудын тухай судлах	08/11 - 08/15	дууссан	✓
2	Жава, Spring Boot, Spring MVC технологиудын тухай судлах	08/11 - 08/15	дууссан	✓
3	Auftragsverwaltung системийн классын диаграмыг гаргах	08/12 - 08/14	дууссан	✓
4	Auftragsverwaltung систем дээр ашигласан зохиомжийн үлгэр загваруудыг олж тогтох	08/14 - 08/18	дууссан	✓
5	МУИС-ын дипломын ажлыг удирдах системийн шаардлагатай танилцах	08/18	дууссан	✓
6	Үг шаардлагын дагуу системийн зохиомжийг загварчлах	08/18 - 08/20	дууссан	✓
7	Системийн хэрэгжүүлэлтийг Жава технологи ашиглан гүйцэтгэх	08/20 - 08/22	дуусаагүй	✓
8	Банкны Excel хуулыг боловсруулах модулийн шинжилгээг гүйцэтгэх	08/22	дууссан	✓
9	Үг шаардлага дээрээ үндэслэн зохиомж болон архитектурыг гаргах	08/23 - 08/24	дууссан	✓
10	Үг зохиомжийн дагуу модулийн хэрэгжүүлэлтийг Spring Boot фреймворк ашиглан гүйцэтгэх	08/25 - 08/29	дууссан	✓
11	Модулийг JUnit санг ашиглан тестлэх	08/27 - 08/30	дууссан	✓

Баталсан: Албан байгууллагын дадлага удирдагч: Х. Ганзориг /.../

Төлөвлөгөө боловсруулсан:

Ц. Биндэрцэцэг /.../ 

11102/0749 д. 5720007

ГАРЧИГ

УДИРТГАЛ	1
БҮЛГҮҮД	2
1. БАЙГУУЛЛАГЫН ТАНИЛЦУУЛГА	2
1.1 Товч танилцуулга	2
1.2 Ямар үйлчилгээ үзүүлдэг вэ?	2
1.3 Ямар систем дээр голчлон төвлөрдөг вэ?	2
2. ЗОРИЛГО БА ЗОРИЛТ	3
2.1 Зорилго	3
2.2 Зорилт	3
3. ОНОЛЫН СУДАЛГАА	4
3.1 Программ хангамжийн зохиомжийн зарчмууд	4
3.2 Программ хангамжийн зохиомжийн үлгэр загварууд: Байгуулалтын, Бүтцийн, Зан байдлын	5
4. АШИГЛАСАН ТЕХНОЛОГИ	13
4.1 Жава технологи	13
4.2 Spring Boot	13
4.3 Tomcat вэб сервер	14
4.4 JPA (Java Persistence API)	14
5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ	16
5.1 Урвуу инженерчлэл: "Auftragsverwaltung" систем дэх зохиомжийн үлгэр загварууд	16
5.2 МУИС-ийн дипломын ажлыг удирдах системийн зохиомж дахь үлгэр загваруудын хэрэглээ	26
6. ДАДЛАГЫН ЯВЦ	28

6.1 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн шинжилгээ	28
6.2 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн зохиомж	38
6.3 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн хэрэгжүүлэлт	42
7. ДҮГНЭЛТ	59
7.1 Yр дүн	59
7.2 Yр дүнгийн тайлан	60
НОМ ЗҮЙ	61
ХАВСРАЛТ	62
A. НЭРШЛИЙН КОНВЕНЦ	62
B. БАНКНЫ ХУУЛГЫН ФОРМАТ	64
C. БУУЛГАЛТЫН ЗАГВАР	65

ЗУРГИЙН ЖАГСААЛТ

5.1	”Applicationlogic” багцын классын диаграм	18
5.2	”Utility” багцын классын диаграм	19
5.3	”DAO” багцын классын диаграм	23
5.4	Дипломын ажлыг удирдах системийн классын диаграм	27
6.1	MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн ажлын явцын диаграм	30
6.2	Excel хуулга боловсруулах ерөнхий төлөвийн диаграм	38
6.3	”Builder” үлгэр загварын зохиомжийн диаграм	39
6.4	MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн model багцын классын диаграм	40
6.5	MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн зохиомжийн үеийн классын диаграм	41
7.1	Гүйлгээний мэдээллүүдийг өгөгдлийн санд амжилттай оруулсан байдал	59
B.1	Хаан банкны гүйлгээний Excel хуулгын формат	64

ХҮСНЭГТИЙН ЖАГСААЛТ

6.1	Ажлын явц: Банкны хуулга оруулах (UC-FS-01)	31
6.2	Ажлын явц: Банкны загвар үүсгэх (UC-FS-02)	32
6.3	Банкны Excel хуулгыг боловсруулах модулийн функциональ шаардлага	34
6.4	Банкны Excel хуулгыг боловсруулах модулийн функциональ бус шаардлага	35
6.5	Банкны Excel хуулгыг боловсруулах модулийн хөгжүүлэлтийн орчинд тавигдах шаардлага	35
6.6	MinuPOS системийн F стандарт бүтцийн толгой баганууд	36
6.7	MinuPOS системийн F стандарт бүтцийн дэлгэрэнгүй баганууд	37
A.1	Auftragsverwaltung систем дээрх Герман-Англи нэршлийн конвенц	62

Кодын жагсаалт

4.1	Spring Boot REST Controller	13
4.2	Spring Boot main class	14
4.3	JPA Entity	15
4.4	Spring Data JPA Repository	15
5.1	Order классын устгагч auftragLoeschen	17
5.2	Order классын арга getAuftragssumme	20
5.3	OrderManagement классын арга hinzufuegen	21
5.4	KundeTO класс	23
6.1	Excel файлаас DataFormatter ашиглан өгөгдлийг шүүж авах	42
6.2	Давхардсан өгөгдлийг шийдвэрлэх	44
6.3	Нөөцийн хэсгээс буулгалтын загваруудын мэдээллийг унших	45
6.4	StatementHdr объект үүсгэх	47
6.5	StatementDetail объект үүсгэх	48
6.6	Өгөгдлийн төрөл хувиргалт	49
6.7	Огнооны хувиргалт	49
6.8	Буулгалтын загвар боловсруулах	51
6.9	БФайлын урьдчилсан харагдац үүсгэх	52
6.10	Classpath-aac буулгалтын JSON-уудыг ачаалах	54
6.11	Файл системээс mapping JSON-уудыг ачаалах	54
6.12	Банкны кодоор mapping-ийг индексжүүлэх	55
6.13	Файлын нэрээр mapping автоматаар сонгох	56
6.14	Runtime reload хийх capability	58
C.1	Хаан банкны гүйлгээний Excel хуулгын форматын буулгалтын загвар	65

УДИРТГАЛ

Миний бие Цэдэн-Ишийн Биндэрцэцэг нь үйлдвэрлэлийн дадлагын З долоо хоногийн хугацаанд программ хангамжийн зохиомжийн үлгэр загваруудын хүрээнд урвуу инженерчлэл -ийн аргачлал, Жава, Spring Boot, Spring MVC гэсэн технологиуд дээр голчлон ажилласан ба уг технологиуд ямар шалтгаанаар үүссэн, хөгжүүлэлтийн ямар арга барил ашигладаг, компаниуд хэрхэн үүн дээр хөгжүүлэлт хийж эцсийн бүтээгдэхүүнийг гаргадаг талаар судлах -ын тулд Java Spring Boot фрэймворк ашигладаг компани болох ”Монгол Ай Ди” байгууллагыг сонгон авч мэргэжлийн дадлагаа гүйцэтгэлээ.

Дадлагын эхний долоо хоногт би бараа захиалгийг зохицуулах ”Auftragsverwaltung” систем дээр урвуу инженерчлэлийн аргачлал ашиглан зохиомжийн үлгэр загваруудыг уг системд хэрхэн ашигласан байгааг олж тогтоосон. Үүний дараа МУИС-ийн дипломын ажлыг удирдах системийн шаардлагийн дагуу системийн зохиомжийг гаргаж, уг зохиомж дээр үндэс -лэн системийг Жава технологи ашиглан хэрэгжүүллээ.

Дадлагын сүүлийн долоо хоногт Монгол Ай Ди компанийн хөгжүүлж буй MinuPOS системийн асуудал шийдвэрлэх хэсэгт ажилласан. Миний хариуцаж авсан модуль нь хэрэглэгчийн дансны Excel хуулгыг сервер рүү ачаалж, хуулга доторх өгөгдлийг боловсруулан системийн өгөгдлийн сантай уялдуулж, хэрэглэгчийн зээлийн мэдээллийг шинэчлэх үүрэгтэй. Уг модуль дээр ажиллахын тулд би Spring Boot, Spring MVC технологийн талаар судалгаа хийж, компанийн хөгжүүлэлтийн арга барилтай танилцаж, өгөгдсөн асуудлыг шийдвэрлэлээ.

1. БАЙГУУЛАГЫН ТАНИЛЦУУЛГА

1.1 Товч танилцуулга

Монгол Ай Ди нь 2015 онд байгуулагдсан Монголын хамгийн анхны албан ёсны төлбөр тооцооны процессорын үйл ажиллагаа эрхлэх зөвшөөрөлтэйгөөр олон улсын стандартын дагуу төлбөр тооцоо, карт, мэдээллийн технологи, лоялти зэрэг чиглэлээр цогц үйл ажиллагаа явуулдаг “Финтек” (Fintech) компани бөгөөд банк санхүү болон ИТ чиглэлийн туршлагатай мэргэжлийн баг бүрэлдэхүүнтэй ба орчин үеийн дэвшилтэт технологийг Монголд хамгийн тэргүүнд нутагшуулан дэлгэрүүлсээр байна.

1.2 Ямар үйлчилгээ үзүүлдэг вэ?

Монгол Ай Ди нь дотоодын болон олон улсын зах зээлд нийцсэн технологиудыг нутагшуулж, блокчэйн, хиймэл оюун ухаан, биометр зэрэг шинэ дэвшилтэт шийдлүүдийг нэвтрүүлэхэд анхаарч ажилладаг. Тус компани нь Монголд анх удаа түлшний төлбөрийн карт, SoftPOS, HCE, DRM зэрэг дэвшилтэт системүүдийг амжилттай нэвтрүүлсэн туршлага -тай. Одоогийн байдлаар зээлийн үйлчилгээ үзүүлдэг.

1.3 Ямар систем дээр голчлон төвлөрдөг вэ?

Монгол Ай Ди нь 2025 оны байдлаар “Mongol ID” систем дээр голчлон төвлөрч байгаа бөгөөд энэ нь хэрэглэгчийн хувийн бүртгэл үүсгэсэнээр олон төрлийн үйлчилгээ, аппликашн (Minu Chat, RedPoint, Minu Pay, Sonos Audiobooks гэх мэт) ашиглах боломж олгодог нэгтгэн төвлө -рүүлсэн платформ юм. Энэ нь санхүүгийн болон санхүүгийн бус олон талын аппликашн, үйлчилгээ рүү нийлэмжтэй холбогддог горим дээр бүтээгдсэн бөгөөд хэрэглэгчид дахин бүртгүүлэхгүйгээр үйлчлүүлэхэд зориулагдсан.

2. ЗОРИЛГО БА ЗОРИЛТ

2.1 Зорилго

Энэхүү үйлдвэрлэлийн дадлагын ажлын хүрээнд программ хангамжийн зохиомжийн үлгэр загваруудыг судалж, тэдгээрийн онцлог, хэрэглээ, давуу болон сул талыг тодорхойлж, бодит төсөл болох ”Банкны Excel хуулга боловсруулах модуль”-ийн тулгамдаж буй асуудлыг шийдэж, Spring Boot фреймворк ашиглан хэрэгжүүлнэ.

2.2 Зорилт

Энэхүү зорилгод хүрэхийн тулд дараах зорилтуудыг тавьж ажиллалаа:

- Программ хангамжийн зохиомжийн үлгэр загваруудын талаар онолын судалгаа хийх;
- ”Auftragsverwaltung” системд ашиглагдсан үлгэр загваруудыг урвуу инженерчлэлийн аргаар илрүүлнэ;
- Онолын судалгааны үр дүнд тулгуурлан МУИС-ийн дипломын ажлыг удирдах системийг зохиомжлоно;
- MinuPOS системийн банкны Excel хуулгыг боловсруулах модульд тулгамдаж буй асуудлыг тодорхойлно;
- Тодорхойлсон асуудал дээр тулгуурлан MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийг зохиомжлоно;
- Spring Boot фреймворк ашиглан MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийг хэрэгжүүлнэ.

3. ОНОЛЫН СУДАЛГАА

Программ хангамжийн зохиомж нь программ хангамжийн системийн бүтэц, түүний бүрэлдэхүүн хэсгүүдийн харилцан үйлчлэл, зан төлөвийг тодорхойлох үйл явц юм. Энэ нь программ хангамжийн хөгжүүлэлтийн амжилтын гол хүчин зүйл бөгөөд системийн чанар, уян хатан байдал, засвар үйлчилгээний чадамжид шууд нөлөөлдөг. Зохиомж нь системийн шаардлага, бизнес-ийн зорилго, техникийн хязгаарлалтыг ойлгох, эдгээрийг зохицуулсан шийдэл гаргахыг шаарддаг.

3.1 Программ хангамжийн зохиомжийн зарчмууд

Программ хангамжийн зохиомжийн зарчмууд нь сайн зохиомжийг бий болгоход чиглэсэн удирдамж, хамгийн сайн туршлагуудын цуглуулга юм. Эдгээр зарчмууд нь программ хангамжийн системийг уян хатан, дахин ашиглах боломжтой, засвар үйлчилгээ хийхэд хялбар болгоход тусалдаг [3]. SOLID зарчмуудыг доор дурдлаа:

- **Single Responsibility Principle:** Класс эсвэл модуль нь зөвхөн нэг л шалтгаанаар өөрчлөгдөх ёстой. Энэ нь кодыг илүү ойлгомжтой, засвар үйлчилгээ хийхэд хялбар болгодог.
- **Open/Closed Principle:** Программ хангамжийн элементүүд нь өргөтгөхөд нээлттэй, өөрчлөхөд хаалттай байх ёстой. Энэ нь шинэ функц нэмэхдээ одоо байгаа кодыг өөрчлөхгүй байхыг шаарддаг.
- **Liskov Substitution Principle:** Суперклассын обьектуудыг түүний дэд классын обьектуудаар орлуулах боломжтой байх ёстой. Энэ нь полиморфизмын үндсэн зарчим бөгөөд кодыг илүү уян хатан болгодог.
- **Interface Segregation Principle:** Том интерфэйсүүдийг жижиг, тодорхой интерфэйсүүд дэд хуваах ёстой. Ингэснээр клиент хэрэгцээгүй аргуудыг хэрэгжүүлэх шаардлагагүй болно.

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

- **Dependency Inversion Principle:** Өндөр түвшний модуль ууд нь доод түвшний модулиудаас бус харин илүү хийсвэр түвшнээс хамааралтай байх ёстой. Энэ нь кодыг уян хатан, дахин ашиглах боломжтой болгодог [3].

3.2 Программ хангамжийн зохиомжийн үлгэр загварууд:

Байгуулалтын, Бүтцийн, Зан байдлын

Зохиомжийн үлгэр загвар нь программ хангамжийн зохиомжийн нийтлэг асуудлуудад бат-лагдсан, дахин ашиглах боломжтой шийдэл юм. Уг нэршлийг 1994 онд "Gang of Four" (GoF) алдаршуулж, хөгжүүлэгчдийн нийтлэг үгсийн сангийн нэг болсон. Ихэнх үлгэр загварыг маш формал байдлаар тайлбарласан байдаг бөгөөд тухайн нөхцөл байдалд тохируулан хуулбар-лаж ашигладаг. Үлгэр загварын тайлбарт ихэвчлэн байдаг хэсгүүд нь:

- **Интент** нь асуудал болон шийдлийн аль алиныг нь товч тайлбарладаг.
- **Хүсэл эрмэлзэл** нь асуудал болон шийдлийг боломжтой болгохыг цааш нь тайлбарладаг.
- **Классуудын бүтэц** нь үлгэр загварын хэсэг бүр, тэдгээр нь хэрхэн уялдаж байгааг харуулдаг.
- Түгээмэл программчлалын хэлнүүдийн нэг дэх **кодын жишээ** нь үлгэр загварын цаад санааг ойлгоход хялбар болгодог.

3.2.1 Байгуулалтын үлгэр загварууд

Байгуулалтын үлгэр загварууд нь уян хатан байдлыг нэмэгдүүлэх, класс болон модулиудыг дахин ашиглахын тулд объект байгуулах ажлыг зохицуулдаг.

"Singleton" үлгэр загвар

Класст зөвхөн нэг тохиолдол байгаа эсэхийг баталгаажуулж, түүнд хандах глобал хандалтын цэгийг үүсгэнэ. Практикт энэ класс нь private байгуулагчтай бөгөөд цорын ганц заагчийг

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

буцаах статик аргатай (эсвэл үүнтэй төстэй) гэсэн үг юм. Энэ үлгэр загвар нь систем дэх үйлдлийг зохицуулахад яг нэг объект шаардлагатай үед хэрэг болно. Гол шинж чанар нь:

- Нэг классын заагч буюу нэг глобал объектыг баталгаажуулдаг.
- Глобал хандалтын цэгээр, жишээ нь getInstance() аргаар хангадаг.
- Ихэвчлэн ”lazy initialization”¹ арга замыг ашигладаг бөгөөд трэдийн аюулгүй байдлын механизмуудыг агуулж болно.

Дундын нөөц эсвэл тохиргоог удирдахад энэ үлгэр загвар ихэвчлэн ашиглагддаг. Жишээлбэл, Java-ийн үндсэн ангиуд java.lang.Runtime болон java.awt.Desktop нь синглтон үлгэр загвараар хэрэгжүүлэгдсэн байдаг.

Давуу тал: Нөөц эсвэл үйлчилгээнд глобал хэмжээнд хандах хандалтыг хялбаршуулдаг. Олон клиент ханддаг байсан ч зөвхөн нэг объект үүсдэг. Мөн lazy initialization нь зөвхөн эхний хэрэглээнд л заагч үүсгэх замаар нөөцийг хэмнэх боломжтой.

Сул тал: Глобал төлөв үүсгэх учир кодыг тестлэхэд хэцүү болгож, далд хамааралтай байдалд хүргэж болзошгүй. Нэг классад заагчийн хяналтыг зохицуулах замаар Single Responsibility Principle зарчмыг зөрчиж байна. Болгоомжтой хэрэгжүүлээгүй тохиолдолд конкурент асуудлууд үүсэж, трэдтэй холбоотой нэмэлт код шаарддаг [3].

”Factory Method” үлгэр загвар

Объект байгуулах интерфэйс эсвэл хийсвэр аргыг тодорхойлдог боловч дэд классуудад аль конкрет классаар объектыг байгуулахыг шийдэх боломжийг олгодог. Үнэн хэрэгтээ объект байгуулах ажлыг үйлдвэрийн дэд классуудад буюу үйлдвэрүүдэд томилон илгээдэг. Энэ нь клиент кодыг конкрет бүтээгдэхүүн классаас салгадаг. Гол шинж чанар нь:

¹”lazy initialization” гэдэг нь объект байгуулах эсвэл утгыг тооцолох ажлыг программыг эхлүүлэх үед биш, харин анх удаа хэрэг болох хүртэл хойшлуулдаг оновчлолын арга зам бөгөөд хэзээ ч ашиглагдахгүй объект байгуулахаас зайлсхийж, программыг эхлүүлэх хугацааг багасгаж, санах ойн ашиглалтыг бууруулдаг.

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

- Үйлдвэрийн интерфэйсийн ард объект байгуулах ажлыг битүүмжилдэг.
- Дэд классууд нь өөр өөр бүтээгдэхүүн буцаахын тулд үйлдвэрийн аргыг дарж бичдэг.
- Клиент нь ямар конкрет төрлөөр бүтээгдсэнийг мэдэхгүйгээр үйлдвэрийн интерфейсийг ашигладаг.
- Клиент кодыг өөрчлөхгүйгээр шинэ бүтээгдэхүүн нэмэх боломжийг олгодог.

Класс нь аль дэд классыг байгуулах ёстойг урьдчилан таамаглах боломжгүй үед энэ үлгэр загвар ашиглагддаг. Жишээлбэл, GUI фреймворк нь платформд тусгайлан зориулсан UI элементүүдийг үүсгэхийн тулд үйлдвэрүүдийг ашиглаж болно. Мөн Жава стандарт сан болон фреймворкуудад өргөн хэрэглэгддэг, жишээ нь Spring, Struts.

Давуу тал: Үйлдвэрийн шинэ дэд классуудыг нэмснээр шинэ бүтээгдэхүүн нэвтрүүлэхэд хялбар. Клиент нь конкрет бүтээгдэхүүн классаас бус зөвхөн үйлдвэрийн интерфейсээс хамаардаг. Ингэснээр байгуулах логикийг бизнесийн логикоос тусгаарлаж өгдөг.

Сул тал: Үйлдвэрийн дэд классуудад нэмэлт класс үүсгэх шаардлагатай бөгөөд энэ нь кодын хэмжээг нэмэгдүүлдэг. Зарим тохиолдолд бүтээгдхүүний төрөл бүрийн хувилбаруудыг дэмжихийн тулд олон үйлдвэр хэрэгтэй болдог [3].

”Abstract Factory” үлгэр загвар

Конкрет классыг тодорхойлохгүйгээр холбоотой эсвэл хамааралтай объектуудын гэр бүлийг үүсгэх интерфейсээр хангадаг. Энэ нь үндсэндээ ”үйлдвэрүүдийн үйлдвэр” юм. Хийсвэр үйлдвэр нь бүтээгдэхүүн тус бүрийг бий болгох аргыг тодорхойлдог бөгөөд конкрет үйлдвэрийн дэд классууд нь конкрет хэрэгжүүлэлтээр хангадаг. Гол шинж чанар нь:

- Хамтдаа ажиллахад зориулагдсан хоорондоо холбоотой объектуудын бүлгүүд буюу бүтээгдэхүүнийг үүсгэдэг.
- Клиентад тодорхой конкрет классаас хамааралгүйгээр бүтээгдэхүүний гэр бүлийг буцаана.

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

- Конкремт үйлдвэр бүр нь гэр бүлийн бүх бүтээгдэхүүнийг бий болгох аргыг хэрэгжүүлдэг.

Аппликешныг олон төрлийн бүтээгдэхүүний аль нэгэнд тохиуулах шаардлагатай үед энэ үлгэр загварыг ашигладаг. Сонгодог жишээ бол олон янзын харагдах байдлыг дэмждэг UI хэрэгсэл юм. AbstractWidgetFactory нь товчлуур, текст хайрцаг, цэс үүсгэж болох ба конкрет үйлдвэрүүд Windows эсвэл Mac загварын хувилбаруудыг үйлдвэрлэдэг.

Давуу тал: Хоорондоо холбоотой бүтээгдэхүүнүүдийг нийцэмжтэй байдлаар хангана. Клиент код нь зөвхөн үйлдвэрийн интерфейстэй ажиллах боломжтой. Мөн үйлдвэрийн дэд классуудыг солилцох замаар өөр өөр бүтээгдэхүүний гэр бүл үүсгэдэг.

Сул тал: Mash олон үйлдвэр, бүтээгдэхүүн классаас хамардаг. Гэр бүлд шинэ бүтээгдэхүүн нэмэхийн тулд хийсвэр интерфейсийг өөрчлөх шаардлагатай нь зохиомжийн Open/Closed зарчмыг зөрчдөг. Конкрет хийсвэр үйлдвэрийн класс нь холбогдох бүтээгдэхүүн бүрийг үйлдвэрлэхийн тулд олон үйлдвэрийн аргын дуудлага ашигладаг [3].

”Builder“ үлгэр загвар

Нарийн төвөгтэй объект байгуулах үйл явцыг алхам алхмаар тусгаарлах боломжийг олгодог. Энэ нь ижил байгуулах үйл явцыг ашиглан өөр өөр дүрслэлийг бий боломжийг олгодог. Гол шинж чанар нь:

- Нарийн төвөгтэй объектуудыг үе шаттайгаар байгуулдаг, жишээ нь олон нэмэлт хэсгүүд эсвэл үүрлэсэн дэд объектуудтай нь хамт.
- *Барилгачин* класс нь эд ангиудыг цуглуулж, эцсийн объектыг build() аргаар угсардаг.
- Захиалагч нь байгуулагчийн интерфэйсээр дамжуулан объект байгуулах үйл явцыг удирддаг. Барилгачин класс нь харин объект үүсгэх ажлыг зохицуулдаг.

Объект нь олон хэсэг эсвэл хослолын сонголтуудыг шаарддаг бөгөөд энэ нь олон тооны байгуулагчуудыг хэт ачаалахад хүргэх нөхцөлд энэ үлгэр загварыг ашигладаг. Жишээ нь, олон сонголттой хоол бүхий хоолны иж бүрдлийг хийх, эсвэл нэмэлт талбар бүхий хэрэглэгчийг

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

байгуулах зэрэг орно. Энэ нь өөрчлөгдөшгүй объектууд эсвэл тохиргооны хүнд объектуудыг (жишээ нь, өгөгдлийн сангийн холболтын тохиргоо, нарийн төвөгтэй GUI бүрэлдэхүүн хэсгүүд) бүтээхэд түгээмэл байдаг.

Давуу тал: Хэд хэдэн аргын дуудалтыг гинжлэх замаар программын уншигдахуйц байдлыг нэмэгдүүлдэг. Байгуулагчын бичдэсийг өөрчлөхгүйгээр нэмэлт параметр эсвэл алхамуудыг хялбархан нэмж болно. Барилгачин объект нь өөрөө түр зуурын, өөрчлөгддөг контейнер бөгөөд эцсийн өөрчлөгдөггүй объектыг бүтээх хүртэл бүх тохиргооны мэдээллийг хадгалдаг.

Сул тал: Үйл явцыг хянахын тулд нэмэлт класс үүсгэх шаардлагатай бөгөөд энэ нь кодын хэмжээг нэмэгдүүлдэг. Бүтээгдэхүүн тус бүрт барилгачин класс хэрэгтэй бөгөөд энэ нь кодын хэмжээг нэмэгдүүлдэг. Энгийн объектуудын хувьд хэт их байж болно [3].

”Prototype” үлгэр загвар

Энэ үлгэр загвар нь бүтээгдэхүүний заагчийг ашиглан шинэ объект үүсгэх боломжийг олгодог. Энэ нь шинэ объект байгуулахын оронд одоо байгаа объектыг хуулбарлах замаар шинэ объект үүсгэдэг. Ингэхдээ конкрет *прототипийг* хэрэгжүүлдэг ихэвчлэн нэг аргатай *clone()* интерфейсийг зарладаг. Клиент нь классуудыг шууд үүсгэхийн оронд прототипийг өөрөө хувилахыг асуух замаар шинэ объектуудыг шаарддаг. Гол шинж чанар нь:

- Объектыг хувилах ажлыг тухайн объектод томилон илгээдэг.
- Кодыг конкрет классуудтай холбооос зайлсхийдэг учир код нь ганц өрөнхий ”prototype” интерфейстэй харьцааг.
- Объект байгуулах нь үнэтэй эсвэл төвөгтэй үед ашигтай байдаг, өөрөөр хэлбэл одоо байгаа прототипийг хуулбарлах нь илүү хялбар байх болно.

Ижил төстэй эсвэл эхнээс нь байгуулахад үнэтэй объектуудыг бүтээхэд хэрэгтэй.

Давуу тал: Клиент хувилах объектын классыг мэдэх шаардлагагүй ба одоо байгаа прототипүүдийг хувилах замаар дахин давтагдах кодыг арилгах боломжтой. Мөн программ ажиллаж байх

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

үед шинэ прототип нэмэхэд хялбар.

Сул тал: Тойрог үүсгэн нэг нэгнээ зааж хандсан эсвэл гадаад нөөцөөс хамаарсан нарийн төвөгтэй объектуудыг хувилах нь төвөгтэй эсвэл алдаатай байж болно. Класс бүрт clone() аргыг хэрэг- жүүлэх ёстой. Гүн ба гүехэн² хуулалттай холбоотой асуудлуудыг анхааралтай авч үзэх шаардлагатай [3].

3.2.2 Бүтцийн үлгэр загварууд

Бүтцийн үлгэр загварууд нь класс болон объектуудыг уян хатан байлгахын зэрэгцээ том бүтэц болгон хэрхэн бүрдүүлэх талаар авч үздэг.

”Composite“ үлгэр загвар

Объектуудыг мод бүтэц болгон зохион байгуулж, нэгж болон бүрэлдэхүүн хэсгүүдтэй ижил байдлаар харьцах боломжийг олгодог. Гол шинж чанар нь:

- Композит болон навчны объектууд ижил интерфэйсийг хэрэгжүүлдэг.
- Композит объект нь хүүхэд объектуудын цуглуулгыг агуулж, хүүхэд объектууд дээр үйлдлүүдийг рекурсивээр гүйцэтгэдэг.
- Клиент нь композит болон навч объектуудтай ижил байдлаар харьцаг.

Ижил төрлийн объектуудын ижил үйлдлийг нэгтгэх шаардлагатай үед энэ үлгэр загварыг ашигладаг. Жишээ нь, график хэрэглэгчийн интерфэйсийн элементүүдийг (жишээ нь, цэс, товчлуур, текст хайрцаг) мод бүтэц болгон зохион байгуулж, хэрэглэгчийн интерфэйсийг ижил байдлаар удирдах боломжийг олгодог. Бидний мэдэх React.js³-ийн бүрэлдэхүүн загвар

²Гүехэн хуулах: Объектын дээд түвшний шинж чанаруудыг хуулах боловч эх хувийн аль ч үүрлэсэн объектыг бус түүний заагчийг оноодог. Гүн хуулах: Шинэ объект үүсгэж, бүх үүрлэсэн объектуудыг давталттайгаар хуулбарлаж, хуулбар нь эх хувилбараас бүрэн хамааралгүй эсэхийг баталгаажуулна.

³<https://react.dev/>

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

болон түүний жижиг бие даасан нэгжүүдийг хослуулан UI-г бий болгох арга нь энэ үлгэр загварын зарчим, давуу талыг шууд тусгасан байdag.

Давуу тал: Композит болон навч объектуудыг ижил байдлаар авч үзэх боломжийг олгодог. Композит бүтэц нь динамикаар өөрчлөгдөж, шинэ хүүхэд объектуудыг нэмэх эсвэл устгах боломжтой. Мөн модны бүтцийн бүх түвшинд үйлдлүүдийг рекурсивээр гүйцэтгэх боломжийг олгодог.

Сул тал: Композит бүтэц нь төвөгтэй бөгөөд удирдах, ойлгоход хэцүү байж болно. Зарим тохиолдолд навч объектуудыг композит объектуудаас ялгах нь төвөгтэй байж болно [3].

”Decorator“ үлгэр загвар

Нэг классын бусад объектод нөлөөлөхгүйгээр тухайн объектод үйлдлийг динамикаар нэмдэг. *Декоратор* нь анхны объектыг баглаж⁴, түүнд ажлыг томilon илгээхээс өмнө/дараа нэмэлт үйлдэл хийх чадамжаар хангадаг. Гол шинж чанар нь:

- Үйлдлийг нэмэх эсвэл өөрчлөхийн тулд тухайн объектын оронд декоратор объект ашигладаг.
- Декоратор класс нь багласан объекттой ижил интерфэйсийг хэрэгжүүлж, зан төлөвийг нэмдэг бөгөөд дараа нь анхны объект руу зурvas дамжуулдаг.
- Олон тооны декоратор объектыг давхарга болгон баглаж болно.

Программ ажиллаж байх үед арга нэмэхэд ихэвчлэн ашиглагддаг. Жишээлбэл, Жава хэлний I/O урсгалууд нь үндсэн InputStream-д аргыг буферлэхийн тулд BufferedInputStream гэх мэт декораторуудыг ашигладаг. GUI кодын хувьд цонхонд гүйлгэх мөр эсвэл хүрээ нэмэхийг мөн декоратороор хийж болно.

⁴”Wrapped object“ гэдэг нь анхны объектын заагчийг агуулсан декоратор бөгөөд ижил интерфейсийг хэрэгжүүлдэг. Энэ нь зурvasыг *baglajc* байгаа анхны объект руу шилжүүлэхээс өмнө эсвэл дараа нь өөрийн логикийг гүйцэтгэх замаар шинэ аргыг нэмж эсвэл одоо байгаа аргыг өөрчилдөг.

3.2. ПРОГРАММ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

Давуу тал: Объектын үйлдлийг динамикаар нэмэх эсвэл өөрчлөх боломжийг олгодог. Анхны класс эсвэл модулийг өөрчлөхгүйгээр шинэ функц нэмэх боломжийг олгодог нь Open/Closed зарчмыг дэмждэг. Мөн олон декораторуудыг рекурсивээр холбож, олон төрлийн үйлдэл хийх чадамжтай объектуудыг бий болгох боломжийг олгодог.

Сул тал: Үр дүнд нь олон жижиг классууд бий болох ба багласан объектыг тестлэх эсвэл дигиталт хийхэд хэцүү байж болно [3].

3.2.3 Зан төлөвийн үлгэр загварууд

Зан төлөвийн үлгэр загварууд нь объект хоорондын холбоос, үүргийг тодорхойлох, хяналтын урсгал болон алгоритмыг удирдах талаар тусгасан байдаг.

Iterator үлгэр загвар

Цуглуулгын элементүүдээр дараалалтайгаар нэвтрэх боломжийг олгодог. Гол шинж чанар нь:

- Цуглуулгын элементүүдээр нэвтрэхийн тулд *итератор* объект ашигладаг.
- Элементүүдээр нэвтрэхийн тулд *next()* болон *hasNext()* аргуудыг хэрэгжүүлдэг.
- Клиент нь зөвхөн итераторын интерфэйстэй харьцааг.

Цуглуулгын элементүүдээр дараалалтайгаар нэвтрэх шаардлагатай үед энэ үлгэр загварыг ашигладаг. Жишээ нь, Java-ийн Collection фреймворк нь List, Set, Map зэрэг цуглуулгуудыг нэвтрэхийн тулд итератор үлгэр загварыг ашигладаг.

Давуу тал: Цуглуулгын элементүүдээр дараалалтайгаар нэвтрэх боломжийг олгодог. Цуглуулгын доторх бүтэц болон төлөвийг далдлах боломжийг олгодог. Мөн олон төрлийн цуглуулгуудыг ижил байдлаар нэвтрэх боломжийг олгодог.

Сул тал: Нэмэлт төвөгтэй байдлыг нэмж оруулдаг. Зарим тохиолдолд итератор нь бүх шаардлагатай өгөгдлийг агуулж чадахгүй байж болно [3].

4. АШИГЛАСАН ТЕХНОЛОГИ

4.1 Жава технологи

Жава нь 1995 онд Sun Microsystems компаниас гарсан, объект хандалтат программчлалын хэл бөгөөд өнөөдөр дэлхий даяар хамгийн өргөн хэрэглэгддэг программчлалын хэлүүдийн нэг юм. Жава технологийг ашиглан вэб, десктоп, мобайл болон сервер талын программ хангамжийг хөгжүүлэх боломжтой. Жава технологийн гол давуу талууд нь платформ хооронд ажиллах чадвар (Write Once, Run Anywhere), хүчирхэг стандарт сангууд, аюулгүй байдал, олон хэрэглэг- чийн дэмжлэг зэрэг юм.

4.2 Spring Boot

Spring Boot нь Жава дээр сууринласан, вэб болон enterprise application хөгжүүлэхэд зориулагдсан фреймворк юм. Spring Boot ашигласнаар Spring-ийн үндсэн тохиргоонуудыг автоматаар хийж өгдөг тул хөгжүүлэгчид богино хугацаанд, цэвэр код бичиж, хурдан хөгжүүлэлт хийх боломжтой. Spring Boot нь REST API, MVC архитектур, security, data access зэрэг олон чухал боломжуудыг дэмждэг. Жишээ нь, энгийн REST API endpoint дараах байдлаар үүсгэж болно:

```
1 @RestController
2
3 public class HelloController {
4
5     @GetMapping("/hello")
6     public String hello() {
7
8         return "Hello from Spring Boot!";
9     }
10 }
```

Код 4.1: Spring Boot REST Controller

4.3 Tomcat вэб сервер

Apache Tomcat нь Java Servlet болон JSP (JavaServer Pages) технологиудыг дэмждэг, нээлттэй эхийн вэб сервер юм. Tomcat нь вэб программыг ажиллуулах, HTTP request-үүдийг хүлээн авах, боловсруулах үүрэгтэй. Spring Boot болон бусад Java вэб framework-үүд Tomcat-ийг embedded байдлаар ашиглах боломжтой тул deployment хийхэд хялбар болдог. Spring Boot application нь Tomcat embedded байдлаар ажилладаг бөгөөд дараах байдлаар main class-ыг үүсгэнэ:

```

1 @SpringBootApplication
2
3 public class Application {
4
5     public static void main(String[] args) {
6
7         SpringApplication.run(Application.class, args);
8     }
9 }
```

Код 4.2: Spring Boot main class

4.4 JPA (Java Persistence API)

JPA нь Java-д өгөгдлийн сангийн persistent буюу байнгын хадгалалт хийх стандарт API юм. JPA-г ашигласнаар өгөгдлийн сангийн хүснэгтүүдийг объект хэлбэрээр удирдах, CRUD (Create, Read, Update, Delete) үйлдлүүдийг хялбар гүйцэтгэх боломжтой. JPA нь Hibernate, EclipseLink зэрэг олон төрлийн implementation-үүдтай бөгөөд Spring Data JPA нь эдгээрийг Spring framework-тэй хослуулан, өгөгдлийн сангийн үйлдлийг автоматжуулж, хөгжүүлэлтийн хурдыг нэмэгдүүлдэг.

JPA ашигласнаар SQL query бичих шаардлага багасч, өгөгдлийн сангийн бүтэц өөрчлөгдөхөд кодын засвар хийхэд хялбар болдог. Мөн энтити классуудыг annotation ашиглан тодорхойлж, өгөгдлийн сангийн хүснэгтүүдтэй холбох боломжтой. Repository interface-үүд нь

стандарт CRUD үйлдлүүдийг автоматаар дэмждэг тул хөгжүүлэгчид илүү цэвэр, богино код бичих боломжтой.

Жишээ нь, энтити класс болон repository дараах байдлаар үүсгэнэ:

```

1  @Entity
2
3  public class User {
4
5      @Id
6      @GeneratedValue(strategy = GenerationType.IDENTITY)
7
8      private Long id;
9
10     private String name;
11
12     // getter, setter
13 }
```

Код 4.3: JPA Entity

```

1  public interface UserRepository extends JpaRepository<User, Long> {
2 }
```

Код 4.4: Spring Data JPA Repository

Spring Data JPA ашигласнаар өгөгдлийн сангийн хүснэгтүүдтэй ажиллах, хайлт хийх, өгөгдөл хадгалах зэрэг үйлдлүүдийг маш энгийн байдлаар гүйцэтгэх боломжтой болдог. Ингэснээр бизнес логик дээр төвлөрч, хөгжүүлэлтийн үр бүтээмжийг нэмэгдүүлдэг.

5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ

5.1 Урвуу инженерчлэл: "Auftragsverwaltung" систем дэх зохиомжийн үлгэр загварууд

Энэ хэсэгт Жава технологийг ашиглан хэрэгжүүлсэн бараа захиалгийг зохицуулах Герман нэршлийн конвенцтэй "Auftragsverwaltung" систем дээр урвуу инженерчлэлийн аргачлалаар программ хангамжийн зохиомжийн үлгэр загваруудыг уг системд хэрхэн ашигласан байгааг олж тогтооно.

5.1.1 Системийн статик загвар

Системийн классын диаграмыг гаргахын тулд Eclipse IDE дээрх PlantUML¹ программ хангамжийн багажыг ашигласан. Гэвч энэ багаж нь классууд хоорондын холбоосуудыг, тухайлбал бүрдмэл, нийлмэл зэрэг холбоог оновчтой гаргаж чадаагүй. Юуны түрүүнд энэ системийн нэршлийн конвенц нь Герман хэл дээр хийгдсэн учир тодорхой үгсийн Герман–Англи нэршлийн конвенцийг гаргах шаардлага үүссэн. Жишээ нь, "Auftragsverwaltung" нь "OrderManagement" буюу захиалгын удирдлага, "Auftrag" нь Order" буюу захиалга, "Kunde" нь "Customer" буюу харилцагч, "auftragLoeschen" нь "deleteOrder" буюу Order (Aufrag) классын устгагч гэх мэт. Хавсралт В хэсгээс эдгээр орчуулгыг харж болно.

Холбоосуудын төрлийг тодорхойлохын тулд класс тус бүрийн устгагчийн хэрэгжүүлэлтийг ажигласан. Жишээ нь, Order (Aufrag), OrderItem (Aufragsposition) классуудын хувьд Order классын объектийг устгах үед OrderItem классын объектуудаас бүрдсэн вектороор entfernenPos аргын тусламжтай гүйж устгаж байна. Харин Customer (Kunde) классын объектыг устгалгүй үлдээсэн байна. Иймд Order, OrderItem классууд хоорондоо нийлмэл холбоотой бол Order, Customer классууд хоорондоо бүрдмэл холбоотой гэж дүгнэлээ.

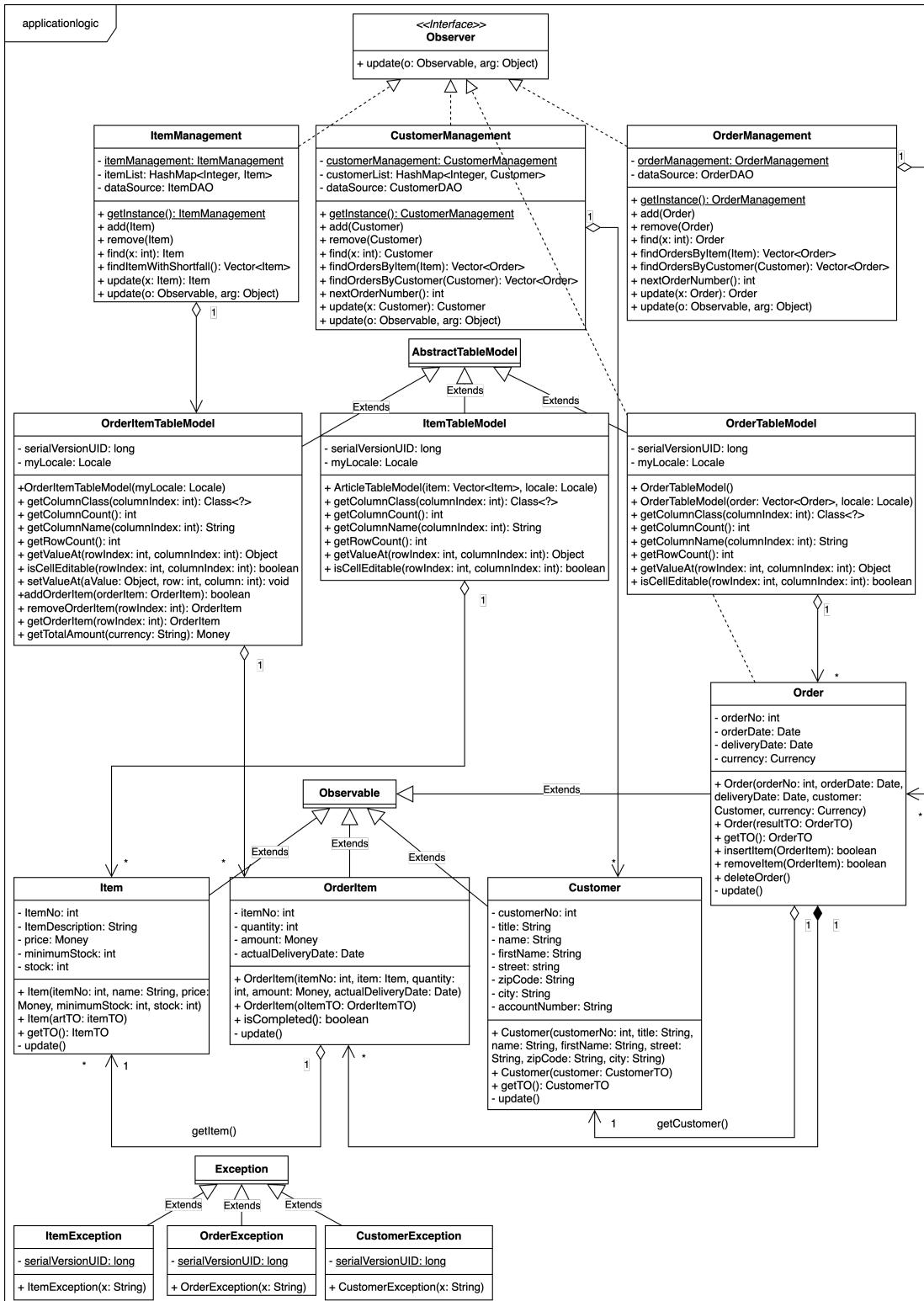
¹<https://plantuml.com/eclipse>

```
1 public boolean entfernenPos(Auftragsposition apos)
2 {
3     boolean rc = apositionen.remove(apos);
4     if (rc)
5         this.aktualisieren();
6     return rc;
7 }
8
9 public void auftragLoeschen()
10 {
11     Iterator<Auftragsposition> positionen =
12         apositionen.iterator();
13     while (positionen.hasNext())
14     {
15         entfernenPos(positionen.next());
16     }
17 }
```

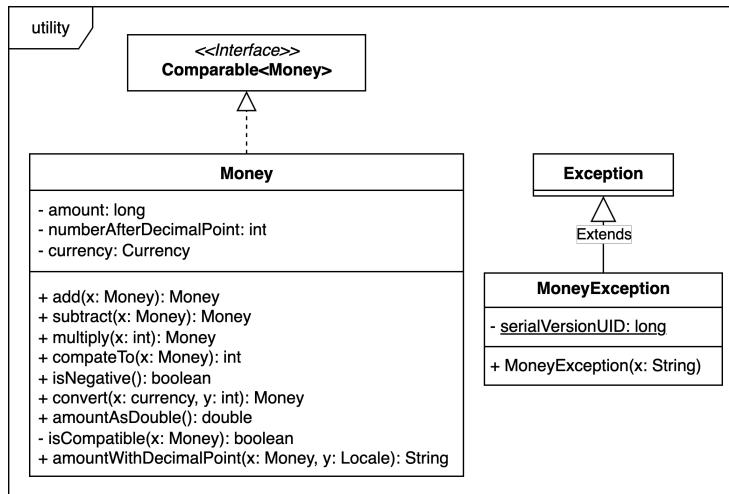
Код 5.1: Order классын устгагч auftragLoeschen

Үүнтэй адилгаар "Applicationlogic" багцын бүх классын нэрсийг жагсааж, тэдгээрийн хоорондын холбоосуудыг илрүүлсэн. Дараа нь, холбоосуудын төрлийг тодорхойлж, UML классын диаграмыг гаргасан. Үүний үр дүнг 5.1 зурагт үзүүлэв. Мөн Item (Artikel) класс нь гадаад "Utility" багцаас (зураг 5.2) Money (Geld) классыг ашиглаж байгаа тул энэ холбоосыг харуулаагүй болно.

5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ
ЗОХИОМЖИЙН УЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ



Зураг 5.1: "Applicationlogic" багцын классын диаграмм



Зураг 5.2: "Utility" багцын классын диаграмм

5.1.2 Системийн зохиомж

Энэ хэсэгт "Auftragsverwaltung" системд илрүүлсэн гол зохиомжийн үлгэр загваруудын зорилго, хэрэглээ, бүтцийн элемент, өмнө тулгарч болох асуудал, үр дүн, жишээ код болон систем доторх бодит хэрэгжүүлэлтийг дэлгэрэнгүй авч үзнэ.

"Iterator" үлгэр загвар

Order классын бүх талбар, аргуудаар статик уншлага хийн `apositionen.iterator()`, `while (positionen.hasNext())`, `positionen.next()` мөрүүдийг тэмдэглэж үлгэр загварт заавал байх элементууд кодод ямар хэлбэрээр илэрсэнг харлаа. Энд `apositionen` нь бүрдэл болж, түүний `iterator()` аргыг дуудаж байгаа нь "Iterator" интерфэйсийг ашиглаж буйг илтгэнэ. `auftragLoeschen()` болон `getAuftragssumme()` мэт аргуудын давталтын логикиг уншиж, хэрхэн давталт явдаг, давталтын үед ямар үйлдэл хийгдэхийг тодруулсан. Давталтын туршид бүрдлүүдтэй хэрхэн харьцаж байгааг анхааран харж, "concurrent modification"² зэрэг

²Конкуррент программчлалд өгөгдлийн бүтцийг өөр процесс эсвэл тредээр давтаж байх үед өөрчилсөн тохиолдолд "concurrent modification" үүсдэг. Энэ нь үнэн байхaa больсон өгөгдлийн бүтцээр гүйснээс үүдсэн урьдчилан таамаглах боломжгүй өгөгдлийн эвдрэл эсвэл "runtime error" алдааг үүсгэж болно.

5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ ЗОХИОМЖИЙН УЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ

асуудал үүсэх эрсдлийг тооцсон. Дээрх ажиглалтаас үлгэр загварын шинж тэмдгүүд илэрсэн тул тухайн кодонд "Iterator" үлгэр загвар ашиглагдсан гэж тодорхойлсон. Энд apositionen нь Vector<Auftragsposition> бөгөөд давталт хийхдээ apositionen.iterator() ашиглагдаж байна. getAuftragssumme() нь while(positionen.hasNext()){...positionen.next()} маягаар стандарт давталтыг ашиглан бүх Amount (Betrag) буюу үнийн дүнг олж байна. Энэ нь "Iterator" интерфэйсийн классик хэрэглээ юм.

```
1 public Geld getAuftragssumme()
2 {
3     Iterator<Auftragsposition> positionen =
4         apositionen.iterator();
5     Geld ergebnis = new Geld(0, 0, waehrung);
6     if (positionen.hasNext())
7         ergebnis = new Geld(positionen.next().getBetrag());
8     while (positionen.hasNext())
9     {
10         ergebnis.addieren(positionen.next().getBetrag());
11     }
12     return ergebnis;
13 }
```

Код 5.2: Order классын арга getAuftragssumme

"Singleton" үлгэр загвар

OrderManagement классын статик талбар дээр шууд new Auftragsverwaltung() дуудаж байгаа нь "eager initialization"³ бөгөөд тредүүдэд аюулгүй байдаг боловч хэзээ ч ашиглагдахгүй объект байгуулагдах эрсдэлтэй. Мөн DAOFactory.getInstance() гэх мэт глобал хандалтын

³"Eager initialization" нь программчлалын стратеги бөгөөд объект эсвэл нөөцийг анх ашиглахыг хүсэх хүртэл хүлээх биш, агуулагдсан класс нь ачаалагдсан даруйд үүсгэгддэг.

**5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ
ЗОХИОМЖИЙН УЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ**

цэгүүдийг ажиглалаа. Класс нь өөрөө "Observer"-ыг хэрэгжүүлж, менежментийн үүрэг гүйцэтгэж байгаа ч байгуулагчийг private гэж огт заагаагүй тул шинэ Auftragsverwaltung объектыг байгуулж параллел объект үүсгэх эрсдэлтэй. hinzufuegen() дотор auf.addObserver(this) гэж байгаа нь Singleton объект өөрөө Observable объектын өөрчлөлтийг хүлээн авч DAO-руу өөрчлөлт илгээж байна. Энэ бүх ажиглалтуудаас жинхэнэ "Singleton" үлгэр загварыг ашиглаагүй гэж дүгнэлээ. Харин өөр нэг үлгэр загварыг олсон нь "Observer" юм. Үүнийг дараагийн хэсэгт тайлбарлав.

```
1  private static Auftragsverwaltung eineAuftragsverwaltung = new
2
3      Auftragsverwaltung();
4
5
6  public void hinzufuegen(Auftrag auf)
7      throws AuftragException
8
9  {
10
11     int auftragsnr = auf.getAuftragsnr();
12
13     try {
14
15         if (datenquelle.read(auftragsnr) != null)
16
17             throw new AuftragException(
18
19                 "Auftrag mit dieser Nummer ist schon vorhanden");
20
21         datenquelle.create(auf.getTO());
22
23         auf.addObserver(this);
24
25     } catch (Exception ex)
26
27     {
28
29         throw new AuftragException("Auftrag " + auftragsnr
30
31             + " konnte nicht gespeichert werden!");
32
33     }
34
35 }
```

Код 5.3: OrderManagement классын арга hinzufuegen

"Observer" үлгэр загвар

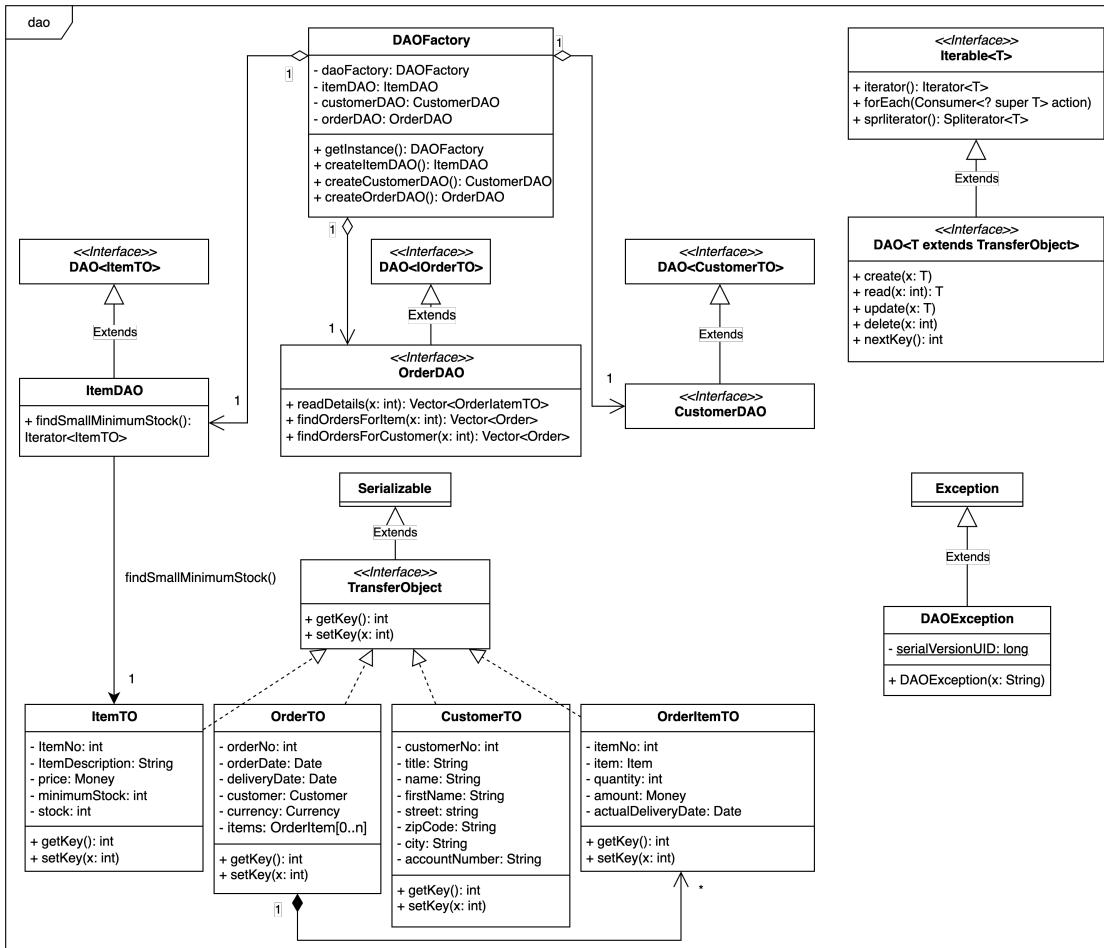
OrderManagement класын `hinzufuegen()` арга дотор `auf.addObserver(this)` гэж байгаа нь Singleton объект өөрөө Observable объектын өөрчлөлтийг хүлээн авч DAO⁴-руу өөрчлөлт илгээж байна. Үүнийг ажигласнаар "Observer" үлгэр загвар ашиглагдсан гэж дүгнэлээ. Домайн объект болох Customer, Order, OrderItem, Item зэрэг классууд нь Observable-ээс удамшиж, өөрийн төлөв өөрчлөгдөх мөчид `notifyObservers()`-ыг дуудаж бүх бүртгэлтэй Observer-уудад мэдээлж байна. CustomerManagement, OrderManagement, ItemManagement зэрэг класс нь Observer интерфейсийг хэрэгжүүлэн, `update()` аргаар домайн обьектоос ирсэн өөрчлөлтийг хүлээн авч байна. Ингэснээр системийн бүх чухал домэйн обьектууд үзэгдлийн төв болж, менежер бүр эдгээр үзэгдлийг сонсож, өөрийн цуглуулгад харгалзах домэйн обьектыг дахин оруулах/шинэчлэх нь төвлөрсөн удирдлага үүсгэхээс гадна домэйн обьект, менежерүү- дийн хооронд нягт уялдаа холбоог бий болгож, нийцэмжтэй байдлыг хадгалж байна.

"Composite" үлгэр загвар

DAO<T extensions TransferObject> интерфэйсийг хэрэгжүүлж болон стандарт CRUD үйлдлүүд (create, read, update, delete) дээр нэмээд `nextKey()`-ийг зарлаж, Iterable<T>-г хүртэл өргөтгөсөн байна. Энэ нь DAO интерфэйсийг хэрэгжүүлсэн класс бүр нь өөрийн төрөлд харгалзах домэйн обьектын цуглуулгыг удирдах үүрэгтэй болохыг илтгэнэ. Жишээ нь, доорх кодонд CustomerDAO нь Customer домэйн обьектын цуглуулгыг удирдаж байна. Үүнийг ажигласнаар "Composite" үлгэр загвар ашиглагдсан гэж дүгнэлээ. Учир нь DAO интерфэйсийг хэрэгжүүлсэн класс бүр нь өөрийн төрөлд харгалзах домэйн обьектын цуглуулгыг удирдах үүрэгтэй бөгөөд эдгээр DAO-уудын цуглуулга нь системийн бүх домэйн обьектын цуглуулгыг бүрдүүлж байна. Энэ нь "Composite" үлгэр загварын үндсэн шинж чанар юм.

⁴"Data Access Object" нь өгөгдлийн нөөцтэй холбоотой бүх үйлдлийг багтааж, хийсвэрлэдэг тусгай давхаргын үүрэг гүйцэтгэдэг ба ын үлдсэн хэсэг нь өгөгдөл хэрхэн, хаана хадгалагдаж байгаагаас хамааралгүй байх боломжийг олгодог.

5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ
ЗОХИОМЖИЙН УЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ



Зураг 5.3: "DAO" багцын классын диаграмм

```

1 public class KundeTO implements TransferObject
2 {
3     ...
4     public KundeTO(int kundennr, String anrede, String name,
5                     String vorname, String ort, String plz,
6                     String strasse, String debitorennr)
7     {
8         this.kundennr = kundennr;

```

**5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ
ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ**

```
9     this.anrede = anrede;
10    this.name = name;
11    this.vorname = vorname;
12    this.strasse = strasse;
13    this.plz = plz;
14    this.ort = ort;
15    this.debitorennr = debitorennr;
16 }
17
18 public int getKey()
19 {
20     return kundennr;
21 }
22 public void setKey(int newKey)
23 {
24     kundennr = newKey;
25 }
26 }
```

Код 5.4: KundеTO класс

5.1.3 Дэд хэсгийн дүгнэлт

Энэ хэсэгт “Auftragsverwaltung” системийг урвуу инженерчлэлийн аргаар задлан шинжилж, статик загвар, нэршлийн конвенци, мөн систем дотор илэрсэн гол зохиомжийн үлгэр загваруудыг тодрууллаа. Ерөнхий ажиглалт нь дараах үндсэн үр дүнг илэрхийлж байна:

- Илрүүлсэн үлгэр загварууд:

- *Iterator*: apositionen.iterator() болон стандарт давталтын логик ашиглагдсан

5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ

нь Iterator үлгэр загварын тод илрэл юм; мөн давталтын явцад concurrent modification эрсдэл бий болж магадгүйг ажигласан.

- *Observer*: Домайн объектууд Observable-ээс удамшиж, менежер класс Observer-ийг хэрэгжүүлэн update() аргыг ашиглаж байсан гэдгээс Observer загвар бодитоор ашиглагдсан.
 - *Singleton (төсмэй)*: Auftragsverwaltung-д eager initialization хэлбэрийн статик талбар байгаа боловч private байгуулагчгүй тул жинхэнэ Singleton гэж дүгнэхэд хүрэлцэхгүй; мөн DAOFactory.getInstance() гэх мэт глобал хандалтын цэгүүд ажиглагдсан.
 - *Composite*: DAO<T extends TransferObject> интерфэйс нь өөрийн төрөлд харгалзах домайн объектын цуглуулгыг удирдаж байсан, энэ нь composite шинжтэй архитектурын хэрэглээ гэж тодорхойлсон.
- **Системийн сайжруулалтын зөвлөмжүүд (ажил хэрэгжүүлэхэд туслах):**
 - Iterator-д объект устгах үед Iterator.remove() эсвэл Copy-on-write⁵ ашиглах замаар concurrent modification-ыг арилгах
 - Singleton үлгэр загвар шаардлагатай бол private байгуулагч болон статик хандагч эсвэл enum-singleton ашиглан жинхэнэ Singleton-г баталгаажуулах; эсвэл глобал статик хандалтыг хязгаарлахын тулд dependency injection ашиглах.

Ерөнхийд нь, урвуу инженерчлэлийн үр дүнд “Auftragsverwaltung” систем нь хэд хэдэн түгээмэл зохиомжийн үлгэр загварыг бодитоор ашигласан болох нь нотлогдов. Гэхдээ зарим хэрэгжүүлэлт нь бүрэн биш ба конкурент, глобал хандалтын менежмент зэрэг талуудыг сайжруулснаар системийн нийцтэй байдал, засвар үйлчилгээ илт сайжрах боломжтой. Энэ дүгнэлтийг үндэслэн дараагийн алхмууд нь кодын цэгцлэлт, synchronization/collection хяналт болон архитектурын жижиг рефакторууд байх ёстой гэж үзлээ.

⁵“Copy-on-write” (CoW) нь өгөгдлийг өөрчлөх хүртэл хуулбарлахыг хойшлуулдаг оновчлолын стратеги юм.

*5.2. МУИС-ИЙН ДИПЛОМЫН АЖЛЫГ УДИРДАХ СИСТЕМИЙН ЗОХИОМЖ
ДАХЬ ҮЛГЭР ЗАГВАРУУДЫН ХЭРЭГЛЭЭ БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ*

5.2 МУИС-ийн дипломын ажлыг удирдах системийн зохиомж дахь

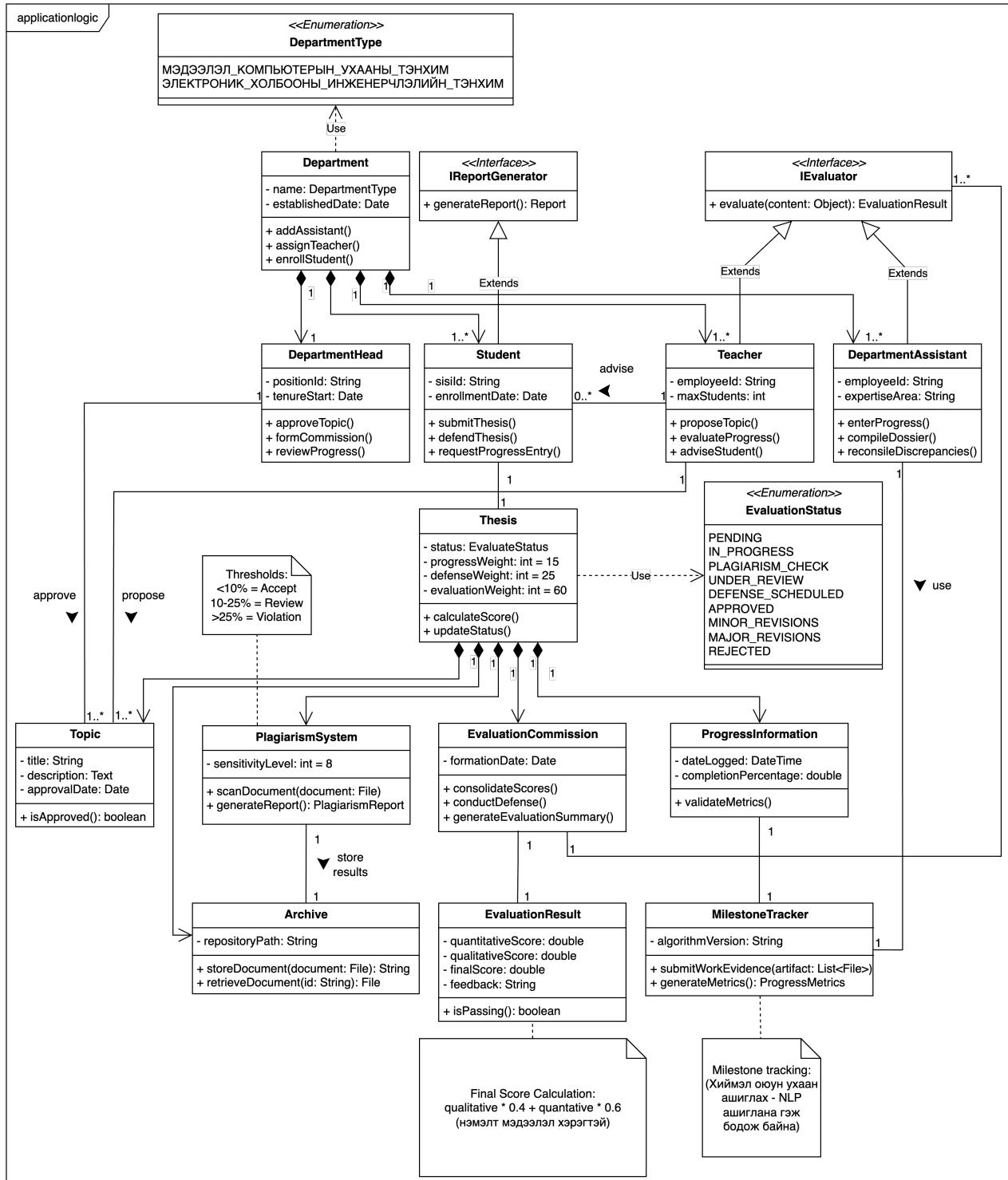
ҮЛГЭР ЗАГВАРУУДЫН ХЭРЭГЛЭЭ

Энэ хэсэгт МУИС-ийн дипломын ажлыг удирдах системийн зохиомжийг дадлага удирдагч Х. Ганзоригийн судалгааны ажилд тусгасан шаардлагын дагуу зохиомжийн үлгэр загварууд, зохиомжийн зарчмуудыг ашиглан гаргасан.

- **Singleton:** DepartmentManager, PlagiarismChecker зэрэг глобал менежерүүдийг нэг экземплярт хязгаарлах зорилгоор private constructor, static хандагчтай ”singleton” үлгэр загварыг ашигласан.
- **Observer:** MilestoneTracker, Student, Supervisor, DepartmentHead зэрэг оролцогчид явцын мэдээлэл, дипломын ажлын статус өөрчлөгдөхөд update мэдэгдэл авах ”observer pattern” үлгэр загварыг ашигласан. Жишээ нь, MilestoneTracker нь дипломын ажилд бүртгэгдэж, явцын өөрчлөлт бүрт метрик тооцоолон тайлан гаргадаг байна.
- **Composite:** Department, Committee, StudentGroup зэрэг цуглуулга объектыг бурдмэл бүтэцтэйгээр удирдах зорилгоор ”composite” үлгэр загварыг ашигласан. Жишээ нь, Committee нь гишүүдээс бурдаж, үнэлгээ нэгтгэх, хамгаалалт зохион байгуулах үйлдлийг цогцоор нь гүйцэтгэдэг.
- **Factory:** ReportGeneratorFactory, EvaluatorFactory зэрэг нь тайлан, үнэлгээний генератор объектыг төрөл бүрээр үүсгэхэд ”factory” үлгэр загварыг ашигласан.
- **Enum, Interface, Abstract class:** Статус, тайлангийн төрөл, milestone зэрэг enum-ууд, IEvaluator, IReportGenerator зэрэг интерфейс, Report зэрэг хийсвэр классуудыг ашиглан системийн өргөтгөх боломж, стандартчиллыг хангаж өгсөн.

Холбоосуудыг UML стандартын дагуу (aggregation, composition, dependency, realization) тэмдэглэсэн. Гэвч одоогоор үлгэр загваруудыг бодит кодын түвшинд хэрэгжүүлээгүй.

**5.2. МУИС-ИЙН ДИПЛОМЫН АЖЛЫГ УДИРДАХ СИСТЕМИЙН ЗОХИОМЖ
ДАХЬ УЛГЭР ЗАГВАРУУДЫН ХЭРЭГЛЭЭ БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ**



Зураг 5.4: Дипломын ажлыг удирдах системийн классын диаграмм

6. ДАДЛАГЫН ЯВЦ

Энэ хэсэгт Монгол Ай Ди компанийн хөгжүүлж буй MinuPOS системийн зээлийн хэсэгт харьяалагдах банкны Excel хуулгыг боловсруулах модуль болон түүнтэй холбогдох асуудлыг шийднэ.

6.1 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн шинжилгээ

Тус модуль нь банкнаас ирсэн Excel файлыг автоматаар уншиж, өгөгдлийг шалган, шаардлагатай тохиолдолд алдааг илрүүлж, мэдээллийг MinuPOS системийн өгөгдлийн санд шууд хадгалах боломжийг бүрдүүлнэ.

6.1.1 MinuPOS системийн танилцуулга

MinuPOS нь бизнес эрхлэгчид болон жижиг дунд аж ахуйн нэгжүүдийн санхүүгийн хэрэгцээг хангах цогц зээлийн систем бүхий платформ юм. Эдгээр зээлийн үйлчилгээнүүдийг MinuPOS-ийн мобайл аппликашн болон POS системээс шууд удирдах боломжтой бөгөөд ингэснээр зээлийн хүсэлт гаргах, төлбөрийн түүхээ харах, POS орлоготой уялдуулан зээлийн төлөлтөө автоматжуулах зэрэг үйлдлийг цахимаар хийх боломжтой. MinuPOS нь Монголын 12 банкны санхүүгийн системтэй шууд холбогдож, хэрэглэгчдийн банкны хуулгыг автоматаар татаж авч, тэдгээрийг боловсруулан зээлийн мэдээлэлтэй уялдуулдаг. Энэ нь хэрэглэгчдэд цаг хугацаа хэмнэх, гар ажиллагааг багасгах, мөн зээлийн эрсдэлийг бууруулахад тусалдаг. MinuPOS нь мөн QR кодын төлбөрийн системийг дэмждэг бөгөөд энэ нь хэрэглэгчдэд хурдан, аюулгүй төлбөр хийх боломжийг олгодог.

6.1.2 Одоогийн MinuPOS системийн банкны Excel хуулгыг боловсруулах модульд тулгарч буй асуудал

Монголын 12 банк тус бүр өөрийн гэсэн Excel хуулгын форматыг ашигладаг тул эдгээр форматуудыг зөв таньж, боловсруулах нь төвөгтэй байдаг. Иймд одоогийн MinuPOS системийн банкны Excel хуулгыг боловсруулах модульд формат бүрт зориулсан урт хэмжээтэй өөр өөр код бичигдсэн байгааг сайжруулах хэрэгтэй байна. Мөн зарим банкны Excel хуулгын форматууд нь цаг хугацааны явцад өөрчлөгдж, шинэчлэгддэг тул эдгээр өөрчлөлтүүдийг код дээр гар аргаар засварлах шаардлагатай болж, энэ нь алдаа гарах магадлалыг нэмэгдүүлдэг. Түүнчлэн зарим тохиолдолд хэрэглэгчид буруу форматтай эсвэл шаардлага хангахгүй Excel файлуудыг оруулдаг тул эдгээр алдааг илрүүлж, хэрэглэгчдэд ойлгомжтой мэдээлэл өгөх механизм дутагдалтай байна.

6.1.3 Системийн орчин

MinuPOS системийн системийн сервер талын код нь Жава технологи дээр бичигдсэн ба үндсэн фреймворк нь Spring Boot юм. MinuPOS нь SoftPOS шийдэлтэй; ухаалаг утсаар карт/NFC төлбөр хүлээн авах боломжийг MineSec SoftPOS¹ -оор хангадаг. MinuPOS системийн худалдан авагч/борлуулагч талын интерфэйс нь POS терминал, SoftPOS, нэгдсэн QR дээр суурилна. Борлуулагч талын Монгол Ай Ди компанийн хөгжүүлсэн iOS/Android мобайл аппууд нийтэд байршсан. MinuPOS систем нь PostgreSQL харьцаат өгөгдлийн санг ашигладаг.

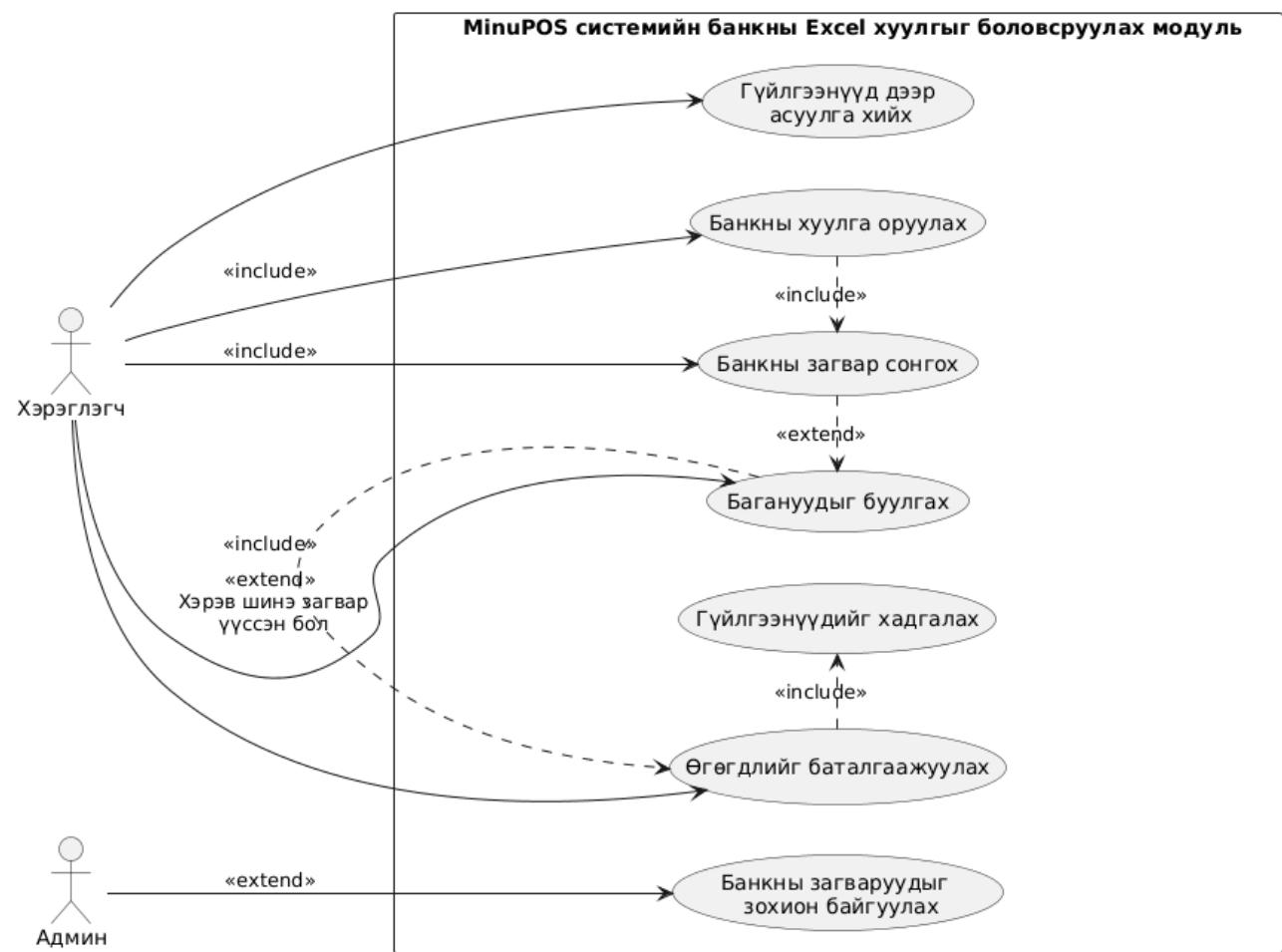
¹<https://minesecsoftpos.com/>

**6.1. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ШИНЖИЛГЭЭ**

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

6.1.4 Динамик загвар

MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн ажлын явцын диаграмыг 6.1 зурагт үзүүлэв.



Зураг 6.1: MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн ажлын явцын диаграмм

Дээрх ажлын явцын диаграммд тусгасан чухал ажлын явцын тайлбаруудыг харгалзах 6.1 болон 6.2 хүснэгтүүдэд үзүүлэв. Энд ажлын явцын өдоөгч үзэгдэл, тоглогч, угтвар нөхцөл, дараах нөхцөл, үр дүн, тайлбар, өргөтгөл, хувилар болон чанарын шаардлагуудыг тус тус харгалзан үзэв.

*6.1. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ШИНЖИЛГЭЭ*

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

Table 6.1: Ажлын явц: Банкны хуулга оруулах (UC-FS-01)

Хэсэг	Агуулга
Тэмдэглэгээ	UC-FS-01
Ажлын явцын нэр	Банкны хуулга оруулах
Ангилал	Анхдагч
Тодорхойлолт	Хэрэглэгч нь Монголын 12 банкны аль нэгээс Excel хуулга оруулж, стандарт хэлбэрт шилжүүлэн хадгалах боломжтой.
Өдөөгч үзэгдэл	Хэрэглэгч веб интерфейсээр файл оруулахыг эхлүүлсэн үед
Тоглогч	Банкны хэрэглэгч, Template Mapping үйлчилгээ, Data Validator, PostgreSQL өгөгдлийн сан
Угтвар нөхцөл	1. Банкны загвар байгаа эсвэл үүсгэж болно.
Дараах нөхцөл	1. Гүйлгээний мэдээлэл өгөгдлийн санд хадгалагдсан байна. 2. Хэрэглэгч үр дүнгийн мэдээлэл авсан байна.
Үр дүн	Стандартчлагдсан санхүүгийн мэдээлэл хэрэглэгчийн атрибуутаар хадгалагдсан байна.
Тайлбар	1. Модуль нь Excel файл болон хэрэглэгчийн сонгосон метадатаг хүлээн авна. 2. Метадатагаас банкны төрлийг тодорхойлно. 3. Баганын буулгалтын загварыг авна. 4. Тохиргооны дүрмээр гүйлгээг задлана. 5. Өгөгдлийн бүрэн бүтэн байдал шалгагдана. 6. Стандарт F хэлбэрт хөрвүүлнэ. 7. Өгөгдлийн санд хадгална.
Өргөтгөл	За. Загвар байхгүй тохиолдолд:

(Үргэлжсилнэ)

*6.1. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ШИНЖИЛГЭЭ*

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

Table 6.1 – Үргэлжлэл

Хэсэг	Агуулга
	<p>За1. Модуль нь баганын буулгалтын загварыг асууна.</p> <p>За2. Хэрэглэгч талбараудыг тодорхойлно.</p> <p>За3. Шинэ загвар үүснэ.</p> <p>5а. Буруу өгөгдөл илэрсэн тохиолдолд:</p> <p>5а1. Хэрэглэгч засаж дахин оруулна.</p>
Онцгой тохиолдол	<p>Өдөөгч: Танигдаагүй файл формат.</p> <p>Өдөөгч: Өгөгдлийн сангийн холболт тасарсан.</p> <p>Өдөөгч: Засаж болохгүй буруу өгөгдөл.</p>
Чанарын шаардлага	<p>QR-01 (Өгөгдлийн үнэн зөв байдал: 99.9%)</p> <p>QR-12 (GDPR нийцтэй байдал)</p> <p>QR-07 (10,000 бичлэгийг <20 секундэд боловсруулна)</p>

Table 6.2: Ажлын явц: Банкны загвар үүсгэх (UC-FS-02)

Хэсэг	Агуулга
Тэмдэглэгээ	UC-FS-02
Ажлын явцын нэр	Банкны загвар үүсгэх
Ангилал	Анхдагч
Тодорхойлолт	Админ нь шинэ банкны хуулгын форматын баганануудын буулгалтыг тодорхойлж, загвар үүсгэнэ.
Өдөөгч үзэгдэл	Шинэ банк нэмэгдсэн эсвэл форматыг өөрчилсөн тохиолдолд
Тоглогч	Админ
Угтвар нөхцөл	1. Жишиг хуулга бэлэн байна.

(Үргэлжилнэ)

*6.1. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ШИНЖИЛГЭЭ*

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

Table 6.2 – үргэлжлэл

Хэсэг	Агуулга
	2. Стандарт талбарууд тодорхойлогдсон байна.
Дараах нөхцөл	1. Шинэ загвар хувилбар хадгалагдсан байна.
Үр дүн	Банкны хуулгад ашиглах дахин ашиглах боломжтой буулгалт бүхий загвар үүссэн байна.
Тайлбар	1. Админ жишиг хуулгыг оруулна. 2. Админ багануудыг стандарт талбаруудад буулгана. 3. Модуль нь буулгалтын бүрэн бүтэн байдлыг шалгана. 4. Загвар хадгалагдана.
Өргөтгөл	4а. Буулгалт гүйцээгүй тохиолдолд: 4а1. Модуль нь дутагдсан талбаруудыг онцолж харуулна. 4а2. Админ нэмэлт буулгалт оруулна. 5а. Өмнөх загвартой зөрчилдөөн гарвал: 5а1. Хувилбаруудын харьцуулалтыг харуулна. 5а2. Админ давхарлах эсэхийг баталгаажуулна.
Онцгой тохиолдол	Өдөөгч: Буруу буулгалтын загвар. Өдөөгч: Хадгалах квота хэтэрсэн.
Чанарын шаардлага	QR-09 (загвар үүсгэх хугацаа < 5 минут) QR-14 (Хувилбар зөрчилоөс урьдчилан сэргийлэх)

*6.1. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ШИНЖИЛГЭЭ*

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

6.1.5 Функциональ шаардлага

Доор MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн функциональ шаардлагуудыг жагсаав. Дадлага удирдагчийн заавраар хэрэглэгч нэвтрэх/бүртгэх шаардлагыг орхигдуулсан болно. Хэрэглэгч нь админ байж болно.

Table 6.3: Банкны Excel хуулгыг боловсруулах модулийн функциональ шаардлага

Шаардлагын нэр	Шаардлагын тайлбар
ФШ10	Модуль нь банкны Excel хуулгаас гүйлгээний мэдээллийг (огноо, дүн, тайлбар) уншиж авна.
ФШ11	Модуль нь өгөгдсөн банкны Excel хуулгаас уншиж авсан мэдээллийг F стандарт бүтэцтэй объектод хувиргана. F стандарт бүтцийн дэлгэрэнгүйг ”Өгөгдлийн бүтэц ба загвар” хэсгээс харна уу.
ФШ12	Модуль нь өгөгдсөн банкны Excel хуулгын мэдээллийг өгөгдлийн санд хадгалдаг байна.
ФШ13	Модуль нь хадгалсан гүйлгээнүүд дээр огноо, дүн, тайлбар зэрэг атрибуутуудаар хайлт хийх боломжийг олгодог байна.
ФШ20	Модуль нь хэрэглэгчээс банкны төрлийг оролтоор авдаг байна.
ФШ21	Модуль нь хэрэглэгчид банкны Excel хуулгын загварыг оролтоор оруулах боломжийг олгодог байна.
ФШ30	Модуль нь хэрэглэгчээс банкны Excel хуулгыг оролтоор авдаг байна.
ФШ40	Модуль нь хэрэглэгчид банкны загвар үүсгэх боломжийг олгодог байна.
ФШ41	Модуль нь хэрэглэгчид хадгалсан банкны загваруудыг засах, устгах боломжийг олгодог байна.

*6.1. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ШИНЖИЛГЭЭ*

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

6.1.6 Функциональ бус шаардлага

Table 6.4: Банкны Excel хуулгыг боловсруулах модулийн функциональ бус шаардлага

Шаардлагын нэр	Шаардлагын тайлбар
ФШБ10	1000 хүртэлх мөр бүхий Excel хуулгыг 5 секундын дотор боловсруулж, өгөгдлийн санд амжилттай хадгалдаг байна.
ФБШ20	Хэрэглэгчийн Excel хуулга оруулах үйлдэл нь ойлгомжтой, 3-аас илүүгүй алхмаар хийгддэг байна.
ФБШ40	Модуль нь шинэ банкны төрөл эсвэл загварыг нэмэхэд хялбар, үүнийг хийхэд одоо байгаа кодонд их хэмжээний өөрчлөлт хийх шаардлагагүй байна.
ФБШ41	Код нь цэгцтэй, тайлбар бичигдсэн, өөр хөгжүүлэгч тухайн кодыг ойлгоход хялбар байна.

6.1.7 Хөгжүүлэлтийн орчинд тавигдах шаардлага

Table 6.5: Банкны Excel хуулгыг боловсруулах модулийн хөгжүүлэлтийн орчинд тавигдах шаардлага

Шаардлагын нэр	Шаардлагын тайлбар
ХОТШ10	Модуль хөгжүүлэхэд зориулсан нэгдсэн хөгжүүлэлтийн орчинг (IDE) ашиглана.
ХОТШ20	Git зэрэг хувилбарын хяналтын системүүдийг ашигладаг байна.
ХОТШ30	Өгөгдлийн хадгалахдаа PostgreSQL харьцаат өгөгдлийн санг ашиглана.

6.1.8 Өгөгдлийн бүтэц ба загвар

Банкны Excel хуулгыг боловсруулах модуль нь F стандарт бүтэктэй өгөгдлийн загварыг ашиглана. F стандарт бүтэц нь дараах төрөл бүхий талбарыг агуулна:

Банкны хуулгын өгөгдлийн баганууд

Table 6.6: MinuPOS системийн F стандарт бүтцийн толгой баганууд

Баганы нэр	Өгөгдлийн төрөл
STATEMENT_ID	VARCHAR2(20)
REQUEST_ID	VARCHAR2(20)
BANK_CODE	VARCHAR2(20)
START_DATE	DATE
END_DATE	DATE
TXN_COUNT	NUMBER
INCOME	NUMBER
OUTCOME	NUMBER
FILE_ID	VARCHAR2(100)
ACCOUNT_NO	VARCHAR2(20)
CREATED_DATE	DATE
STATEMENT_DATE	DATE
MAIN_FLAG	VARCHAR2(2)
ACCOUNT_NAME	VARCHAR2(200)

Банкны хуулгын гүйлгээний дэлгэрэнгүй баганууд

Table 6.7: MinuPOS системийн F стандарт бүтцийн дэлгэрэнгүй баганууд

Баганы нэр	Өгөгдлийн төрөл
DETAIL_ID	VARCHAR2(20)
STATEMENT_ID	VARCHAR2(20)
TXN_DATE	DATE
PRE_BALANCE	NUMBER
POST_BALANCE	NUMBER
TXN_AMOUNT	NUMBER
TXN_TYPE	VARCHAR2(20)
TXN_DESC	VARCHAR2(1000)
CO_ACCOUNT	VARCHAR2(200)
USE_FLAG	VARCHAR2(2)
USE_FLAG_USER_ID	VARCHAR2(20)
USE_FLAG_DATE	DATE
STATUS	VARCHAR2(20)

Дадлагын ажлын хүрээнд миний гүйцэтгэх даалгавар нь асуудал шийдэх байсан тул шинжилгээний үеийн классын диаграмыг үүсгээгүй болно. Гэхдээ статик загварыг доорх зохиомжийн хэсэгт дэлгэрэнгүй тайлбарлав.

6.2 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн зохиомж

Энэ хэсэгт MinuPOS системийн банкны Excel хуулгыг боловсруулах модульд ашигласан гол архитектурын үлгэр загваруудыг танилцуулж, тэдгээрийн хэрэглээ, зохиомжийн тайлбарыг өгнө.

6.2.1 "Strategy" үлгэр загварын зохиомжийн тайлбар

Монголын банкууд Excel хуулгаа өөр өөр форматтайгаар өгдөг тул формат бүрт тусдаа, хатуу кодчилсон парсер бичих нь үр ашиггүй. "Strategy" үлгэр загварыг сонгосон гол шалтгаан нь шинэ банк нэмэх, эсвэл хуучин банкны хуулгын формат өөрчлөгдөхөд үндсэн парсингийн логикт өөрчлөлт хийх шаардлагагүй болоход оршино. Зөвхөн шинэ буулгалтын загвар нэмэхэд хангалттай. Ингэснээр код давхардал багасгаж чадна.

6.2.2 "Template Method" үлгэр загварын зохиомжийн тайлбар

Excel хуулга боловсруулахад "Template Method" үлгэр загварыг ашигласан. Ямар ч банкны Excel файлын хувьд ерөнхий алгоритм дараах алхмуудтай ижил байна:

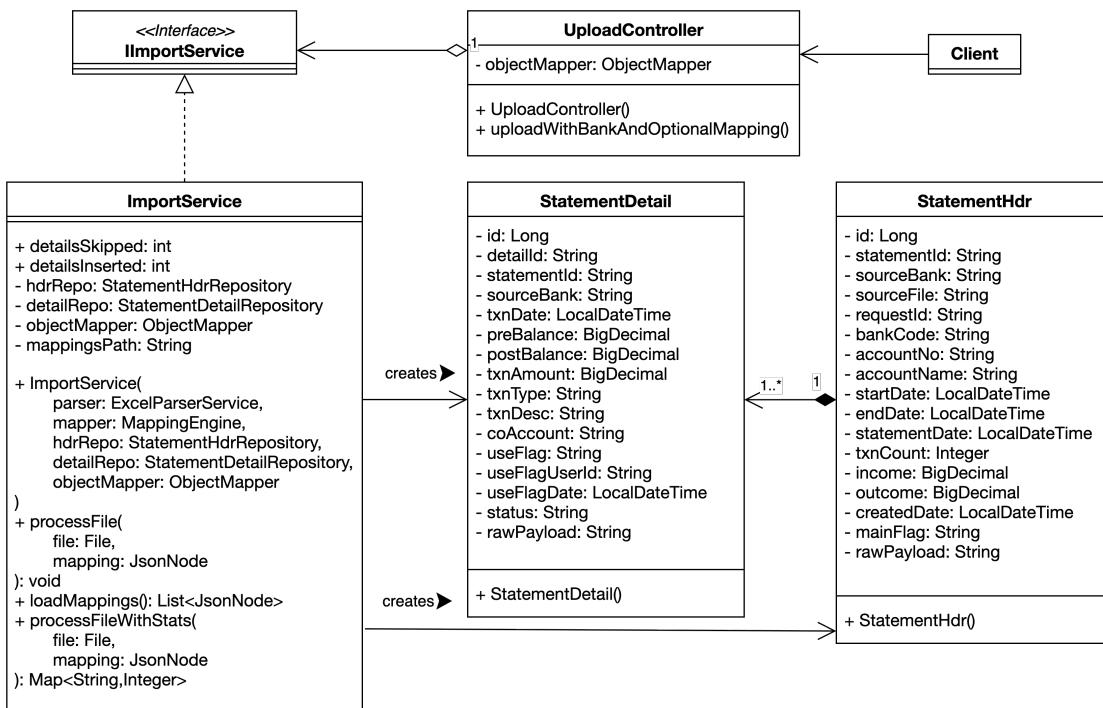


Зураг 6.2: Excel хуулга боловсруулах ерөнхий төлөвийн диаграм

Энэ урсгал тогтмол боловч зарим алхам (жишээлбэл, багануудыг хэрхэн буулгах) нь банк бүрийн форматаас хамааран өөрчлөгдж болно. Үүний тулд бүх хувилбар тус бүрээр дахин код бичихийн оронд өрөнхий алгоритмыг нэг загварт хадгалж, хувьсах алхмуудыг буулгалтын загвараар уян хатан болгож чадна.

6.2.3 "Builder" үлгэр загварын зохиомжийн тайлбар

Модуль тодорх StatementHdr, StatementDetail зэрэг энтити объектуудыг үүсгэхэд "Builder" үлгэр загварыг ашигласан (6.3 зургийг харна уу). Эдгээр энтитинүүд нь олон талбартай бөгөөд бүх утгууд нь нэг дор бэлэн байдаггүй. Барилгачин нь урт байгуулагч бичихгүйгээр, эсвэл бүх талбарыг гараар тохируулахгүйгээр объектуудыг алхам алхмаар үүсгэх боломжтой болгодог. Ирээдүйд шинэ талбар нэмэх шаардлага гарвал байгуулагч бүрийг өөрчлөхгүй, зөвхөн барилгачинд нэмэхэд хангалттай. (6.3 зургийг харна уу)



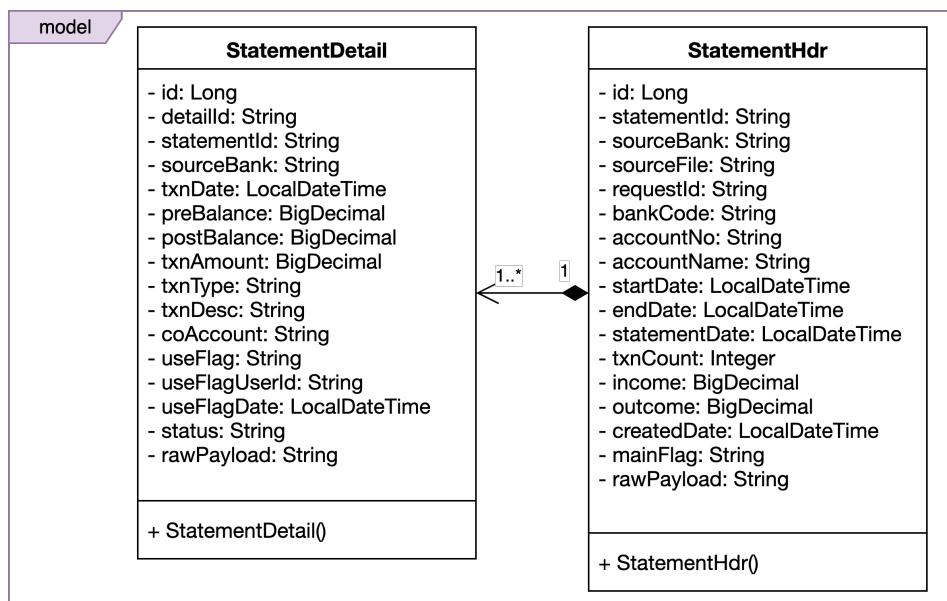
Зураг 6.3: "Builder" үлгэр загварын зохиомжийн диаграм

6.2.4 "Factory" үлгэр загварын зохиомжийн тайлбар

"Factory" үлгэр загварыг MappingConfigLoader классад тусгасан. Энэ класс нь буулгалтын загвар зэрэг объектуудыг үүсгэх үүрэгтэй бөгөөд тухайн объектыг хэрхэн бүтээх нарийн логикийг системийн бусад хэсэгт ил гаргахгүй. Жишээлбэл, JSON буулгалтын загварыг уншиж, тохирох буулгалтын объект үүсгэх ажлыг хариуцна.

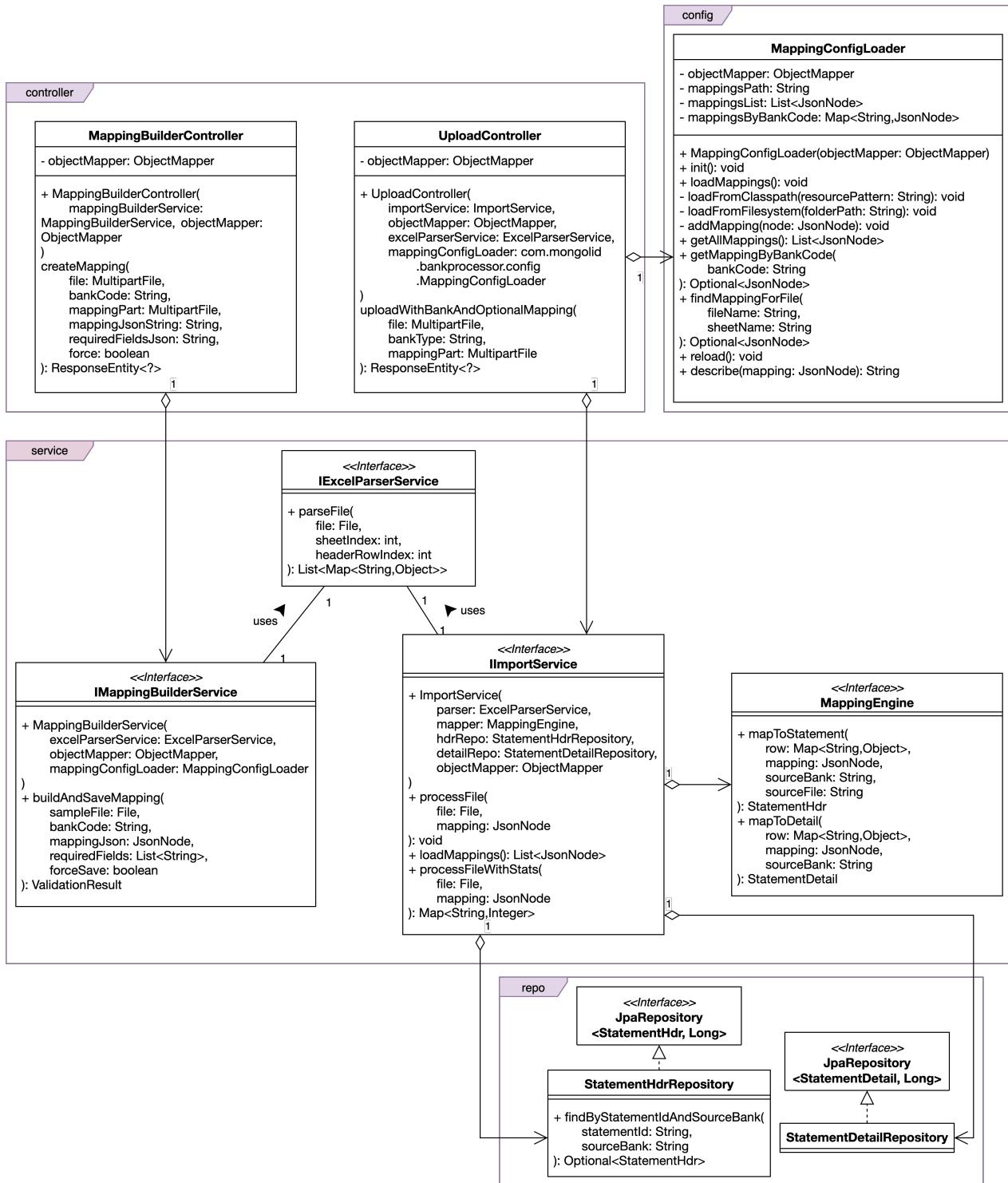
6.2.5 Модулийн статик загвар

Шинжилгээний үед тодорхойлсон өгөгдлийн F стандарт бүтэцтэй нийцэхээр model багцын классын диаграммыг 6.4 зурагт үзүүлэв. Мэдээж getter, setter, toString зэрэг агуудыг агуулах бөгөөд эдгээрийг диаграмд оруулаагүй болно. Модулийн гол зохиомжийн диаграммыг 6.5 зурагт үзүүлэв.



Зураг 6.4: MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн model багцын классын диаграмм

6.2. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ МОДУЛИЙН ЗОХИОМЖ



Зураг 6.5: MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн зохиомжийн ўеийн классын диаграм

6.3 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн хэрэгжүүлэлт

6.3.1 ExcelParserService

Файлын форматын олон янз байдлыг дэмжихийн тулд зөвхөн өгөгдсөн буулгалтын загвар дээр тулгуурлан Excel файлаас өгөгдлийг шүүж авдаг байхаар хэрэгжүүлсэн. Жишээлбэл, Excel файлын толгой мөрийн индексийг зааж өгсөн тохиолдолд тухайн мөрөөс баганын нэrsийг задлан, дараагийн мөрүүдийг боловсруулахад ашигладаг. Мөн FORMULA төрлийн нүднүүдийн бодит утгыг авах, NUMERIC төрлийн нүднүүдийг тоо эсвэл огноо байдлаар ялгаж боловсруулах зэрэгт DataFormatter классыг ашигласан.

POI-ийн "Cannot get a NUMERIC value from a STRING cell" алдааг засахын тулд CellType-ыг шалгаж, зөвхөн NUMERIC үед getNumericCellValue() эсвэл getDateCellValue() дуудаж, бусад тохиолдолд DataFormatter-aap string утгыг авдаг болгосон. Харин FORMULA төрлийн хувьд getCachedFormulaResultType() ашиглаж, row.getLastCellNum()-ийн давталт дотор < нөхцөл тавьсан. Мөн mapToDetail болон mapToStatement хэсэгт String утгыг Double эсвэл Date рүү parse хийдэг хэсэг нэмсэн.

```
1 for (int ci = 0; ci < lastCellNum; ci++) {  
2     Cell cell = row.getCell(ci);  
3     String key = headerMap.containsKey(ci) ? headerMap.get(ci) : String  
4         .valueOf(ci);  
5     Object val = null;  
6     if (cell != null) {  
7         CellType cellType = cell.getCellType();  
8         if (cellType == CellType.FORMULA) {  
9             cellType = cell.getCachedFormulaResultType();
```

```
9
10
11     if (cellType == CellType.NUMERIC) {
12
13         if (DateUtil.isCellDateFormatted(cell)) {
14
15             val = cell.getDateCellValue();
16
17         } else {
18
19             val = cell.getNumericCellValue();
20
21         }
22
23
24     if (val != null && !(val instanceof String && ((String) val).
25
26         isEmpty())) {
27
28         allEmpty = false;
29
30     }
31
32     rowMap.put(key, val);
33
34 }
```

Код 6.1: Excel файлаас DataFormatter ашиглан өгөгдлийг шүүж авах

6.3.2 ImportService

ImportService нь Excel файлыг уншиж, буулгалтын загвар ашиглан өгөгдлийг стандарт форматад хувиргаж, өгөгдлийн санд хадгалах гол үйлчилгээ юм. ExcelParserService-ашиглан

файлыг задлах, MappingEngine-ээр өгөгдлийг стандарт форматад хувиргах, Repository-гүүдээр өгөгдлийн санд хадгалах, мөн буулгалтын загваруудыг JSON хэлбэрээр нөөцөөс унших зэрэг үүргүүдтэй. Уг үйлчилгээний гол алхмуудыг дараах байдлаар нэгтгэн тайлбарлав:

- Буулгалтын загвар байхгүй тохиолдолд файлыг боловсруулахгүй.
 - Excel-ийн мөрүүдийг STATEMENT_ID талбараар ялгаж, толгой өгөгдөл болон дэлгэрэнгүй өгөгдлийг тодорхойлно.
 - Ижил StatementID-тай өгөгдөл дахин орж ирвэл шинэчлэнэ, эсвэл шинээр үүсгэнэ.
 - Дэлгэрэнгүй мөрүүдийг холбогдох толгойтой StatementID ашиглан холбоно.
 - Дутуу эсвэл буруу өгөгдлтэй мөрүүдийг автоматаар бөглөж, шаардлагатай тохиолдолд алгасаж, алдааны статистикийг цуглувална.
 - Нөөцөөс mappings/*.json файлуудыг хайж, бүгдийг уншиж JsonNode обьектуудын жагсаалт болгон буцаана.
 - Импортлох явцад statementsInserted, statementsUpdated, detailsInserted, detailsSkipped зэрэг статистик үзүүлэлтүүдийг цуглувалж, хэрэглэгчдэд үр дүнгийн мэдээлэл өгнө.

```
1 Optional<StatementHdr> existing = (s.getStatementId() != null) ?  
2     hdrRepo.findByStatementIdAndSourceBank(s.getStatementId(),  
3     sourceBank) : Optional.empty();  
4  
5 if (existing.isPresent()) {  
6     StatementHdr ex = existing.get();  
7     ex.setRawPayload(s.getRawPayload());  
8     ex.setAccountName(s.getAccountName());  
9     ex.setAccountNo(s.getAccountNo());  
10    ex.setStartDate(s.getStartDate());
```

6.3. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ХЭРЭГЖҮҮЛЭЛТ

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

```
10     ex.setEndDate(s.getEndDate());
11
12     hdrRepo.save(ex);
13
14 } else {
15
16     hdrRepo.save(s);
17
18 }
```

Код 6.2: Давхардсан өгөгдлийг шийдвэрлэх

```
1 public List<JsonNode> loadMappings() throws Exception {
2
3     List<JsonNode> list = new ArrayList<>();
4
5     Resource[] resources = new PathMatchingResourcePatternResolver()
6
7         .getResources("classpath*:mappings/*.json");
8
9     for (Resource r : resources) {
10
11         String s;
12
13         try (InputStream in = r.getInputStream()) {
14
15             s = new String(in.readAllBytes(), StandardCharsets.UTF_8)
16
17             ;
18
19         }
20
21         if (s != null && !s.isEmpty()) {
22
23             list.add(objectMapper.readTree(s));
24
25         }
26
27     }
28
29     return list;
30 }
```

Код 6.3: Нөөцийн хэсгээс буулгалтын загваруудын мэдээллийг унших

6.3.3 MappingEngine

MappingEngine класс нь Excel-ээс уншсан Map<String, Object> хэлбэртэй өгөгдлийг MinuPOS системийн F стандарт бүтэцтэй StatementHdr болон StatementDetail объект руу хувиргах үүрэгтэй. Энэхүү үйлчилгээ нь банк бүрийн Excel хуулгын ялгаатай форматыг уян хатан дэмжих, өгөгдлийн төрөл хувиргалт, гүйлгээний дүнг зөв тодорхойлох, алдааны толерант байдал зэрэг гол асуудлуудыг шийддэг. Уг үйлчилгээний гол алхмуудыг дараах байдлаар нэгтгэн тайлбарлав:

- Текст, тоо, огноо зэрэг төрөл бүрийн өгөгдлийг зохих форматад хувиргана.
 - Банкууд гүйлгээний дүнг TXN_AMOUNT, DEBIT/CREDIT, эсвэл баганын дугаар гэх мэт өөр өөрөөр илэрхийлдэг тул бүх боломжит хэлбэрүүдийг шалгаж, нэгдсэн дүн гаргана.
 - Excel-ээс уншсан өгөгдлийг тоо, огноо, текст хэлбэрээс стандартчилна.
 - Excel дотор огноонууд өөр өөр хэлбэрээр хадгалагддаг тул бүх хэлбэрийг стандарт LocalDateTime формат руу хувиргана.
 - Алдаа гарсан тохиолдолд процесс үргэлжилж, алдаатай өгөгдлийг алгасах боломжтой.

MappingEngine болон ImportService-д null-safety хамгаалалт нэмсэн. Ялангуяа mapping, mapping.mapping, statement/detail node-үүд байхгүй үед NPE үүсэхгүй болгов. mapToStatement болон mapToDetail функцүүд null буцааж болох тул ImportService-д үр дүнг шалгаж, null бол хадгалахгүй, алдааны лог бичдэг болгосон. Мөн loadMappings() функц classpath-aac InputStream ашиглан mapping JSON-уудыг зөв уншдаг болсон. Ингэснээр mapping.json дутуу эсвэл буруу үед систем унахгүй, алдааг логлоод үргэлжлүүлдэг болсон.

```
1 public StatementHdr mapToStatement(Map<String, Object> row, JsonNode
2   mapping, String sourceBank, String sourceFile) {
3
4   if (mapping == null || mapping.isMissingNode()) return null;
5
6   JsonNode stmtMap = mapping.get("statement");
7
8   if (stmtMap == null || stmtMap.isNull()) return null;
9
10  StatementHdr s = new StatementHdr();
11
12  s.setSourceBank(sourceBank);
13
14  s.setSourceFile(sourceFile);
15
16  s.setRawPayload(toJson(row));
17
18
19  s.setStatementId(getString(row, stmtMap, "STATEMENT_ID"));
20
21  s.setRequestId(getString(row, stmtMap, "REQUEST_ID"));
22
23  s.setBankCode(getString(row, stmtMap, "BANK_CODE"));
24
25  s.setAccountNo(getString(row, stmtMap, "ACCOUNT_NO"));
26
27  s.setAccountName(getString(row, stmtMap, "ACCOUNT_NAME"));
28
29  s.setMainFlag(getString(row, stmtMap, "MAIN_FLAG"));
30
31  s.setTxnCount.getInt(row, stmtMap, "TXN_COUNT");
32
33  s.setIncome.getDecimal(row, stmtMap, "INCOME");
34
35  s.setOutcome.getDecimal(row, stmtMap, "OUTCOME");
36
37  s.setStartDate.getDateTime(row, stmtMap, "START_DATE");
38
39  s.setEndDate.getDateTime(row, stmtMap, "END_DATE");
40
41  s.setCreatedDate.getDateTime(row, stmtMap, "CREATED_DATE");
42
43  s.setStatementDate.getDateTime(row, stmtMap, "STATEMENT_DATE");
44
45
46  return s;
47}
```

Код 6.4: StatementHdr объект уусгэх

```
1 public StatementDetail mapToDetail(Map<String, Object> row, JsonNode
2     mapping, String sourceBank) {
3
4     if (mapping == null || mapping.isMissingNode()) return null;
5
6     JsonNode dmap = mapping.get("detail");
7
8     if (dmap == null || dmap.isNull()) return null;
9
10    StatementDetail d = new StatementDetail();
11
12    d.setSourceBank(sourceBank);
13
14    d.setRawPayload(toJson(row));
15
16
17    d.setDetailId(getString(row, dmap, "DETAIL_ID"));
18
19    d.setStatementId(getString(row, dmap, "STATEMENT_ID"));
20
21    d.setTxnType(getString(row, dmap, "TXN_TYPE"));
22
23    d.setTxnDesc(getString(row, dmap, "TXN_DESC"));
24
25    d.setCoAccount(getString(row, dmap, "CO_ACCOUNT"));
26
27    d.setUseFlag(getString(row, dmap, "USE_FLAG"));
28
29    d.setUseFlagUserId(getString(row, dmap, "USE_FLAG_USER_ID"));
30
31    d.setStatus(getString(row, dmap, "STATUS"));
32
33
34    d.setTxnDate(getDateTime(row, dmap, "TXN_DATE"));
35
36    d.setUseFlagDate(getDateTime(row, dmap, "USE_FLAG_DATE"));
37
38
39    d.setPreBalance(getDecimal(row, dmap, "PRE_BALANCE"));
40
41    d.setPostBalance(getDecimal(row, dmap, "POST_BALANCE"));
42
43    d.setTxnAmount(pickTxnAmount(row, dmap));
44
45    return d;
46
47 }
```

Код 6.5: StatementDetail объект үүсгэх

```
1 private String normalizeValue(Object v) {
2     if (v == null) return null;
3     if (v instanceof Number) {
4         double d = ((Number) v).doubleValue();
5         if (d == Math.floor(d)) return String.valueOf((long) d);
6         return String.valueOf(d);
7     } else if (v instanceof Date) {
8         return java.time.format.DateTimeFormatter.ISO_LOCAL_DATE_TIME
9             .format(((Date) v).toInstant().atZone(ZoneId.
10                 systemDefault()).toLocalDateTime());
11    } else {
12        return String.valueOf(v).trim();
13    }
14 }
```

Код 6.6: Өгөгдлийн төрөл хувиргалт

```
1 private LocalDateTime getDateTime(Map<String, Object> row, JsonNode
2   mapRoot, String field) {
3
4   try {
5
6     JsonNode node = nodeFor(mapRoot, field);
7
8     if (node == null) return null;
9
10    String key = keyNameFromNode(node);
11
12    Object v = row.get(key);
13
14    if (v == null) return null;
15
16
17    if (v instanceof Date) {
18
19      return LocalDateTime.ofInstant(((Date) v).toInstant(),
```

6.3. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ХЭРЭГЖҮҮЛЭЛТ

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

```
11     ZoneId.systemDefault());
12
13 } else if (v instanceof Number) {
14
15     double d = ((Number) v).doubleValue();
16
17     long epochMillis = (long) ((d - 25569d) * 86400d * 1000d);
18
19     ; // excel serial -> epoch
20
21     return LocalDateTime.ofInstant(Instant.ofEpochMilli(
22
23         epochMillis), ZoneId.systemDefault());
24
25 } else {
26
27     String s = String.valueOf(v).trim();
28
29     if (s.isEmpty()) return null;
30
31     try {
32
33         return LocalDateTime.parse(s);
34
35     } catch (Exception e1) {
36
37         try {
38
39             return LocalDate.parse(s).atStartOfDay();
40
41         } catch (Exception e2) {
42
43             try {
44
45                 java.time.format.DateTimeFormatter fmt = java
46
47                     .time.format.DateTimeFormatter.ofPattern("
48
49                         yyyy-MM-dd HH:mm:ss");
50
51
52             return LocalDateTime.parse(s, fmt);
53
54         } catch (Exception e3) {
55
56             // yyyy-MM-dd
57
58             try {
59
60                 java.time.format.DateTimeFormatter f2 =
61
62                     java.time.format.DateTimeFormatter.
63
64                     ofPattern("yyyy-MM-dd");
65
66             return LocalDate.parse(s, f2).
67
68         }
69
70     }
71
72 }
73
74 }
```

```
33                                         atStartOfDay();  
34             } catch (Exception ignore) {  
35                 }  
36             }  
37         }  
38     } catch (Exception e) {  
39         // hhe  
40     }  
41     return null;  
42 }
```

Код 6.7: Огнооны хувиргалт

6.3.4 UploadController

Энэ UploadController класс нь Excel файл болон mapping configuration файлуудыг хүлээн авах, боловсруулах REST API эндпойнтуудыг нэгтгэсэн Spring Boot контроллер юм. Үндсэн үүргүүд нь:

- Excel файлыг Multipart form data хэлбэрээр хүлээн авах.
 - Хэрэглэгчийн өгсөн эсвэл системд бэлэн байгаа буулгалтыг ашиглаж буулгалтын загвар боловсруулах.
 - ImportService-ийг дуудаж импортлох үйл явцыг эхлүүлэх.
 - Импортын статистик мэдээллэлтэй хамт үр дүнг буцаах.

```
1 JsonNode mappingJson = null;  
2 if (mappingPart != null && !mappingPart.isEmpty()) {
```

6.3. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ХЭРЭГЖҮҮЛЭЛТ

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

```
3  tempMappingPath = Files.createTempFile("mapping-", ".json");
4  mappingPart.transferTo(tempMappingPath.toFile());
5  String mappingContent = Files.readString(tempMappingPath);
6  mappingJson = objectMapper.readTree(mappingContent);
7  logger.info("Using user-supplied mapping (mapping file).");
8 } else {
9  Optional<JsonNode> maybe = mappingConfigLoader.getMappingByBankCode(
10    bankType);
11 if (maybe.isPresent()) {
12  mappingJson = maybe.get();
13  logger.info("Using mapping from MappingConfigLoader for bankType={}
14    ", bankType);
15 } else {
16  return ResponseEntity.badRequest().body(Map.of(
17    "error", "No mapping provided and no mapping found for bankType
18      : " + bankType,
19    "hint", "Upload a mapping JSON with the multipart key 'mapping'
20      or add a mapping file to mappings/folder"
21  ));
22 }
23 }
```

Код 6.8: Буулгалтын загвар боловсруулах

Хэрэглэгч буулгалтын файл өгсөн бол түүнийг, үгүй бол системийн буулгалтын загварыг ашиглана.

```
1 int sheetIndex = mappingJson.path("sheet").asInt(0);
2 int headerRowIndex = mappingJson.path("headerRowIndex").asInt(-1);
3 var previewRows = excelParserService.parseFile(tempFilePath.toFile(),
```

```
sheetIndex, headerRowIndex);  
4 int previewCount = previewRows.size();
```

Код 6.9: БФайлын урьдчилсан харагдац үүсгэх

6.3.5 *MappingConfigLoader*

MappingConfigLoader классыг хэрэгжүүлэхэд банк бүрийн Excel хуулгын формат, файл нэр, sheet нэр, баганын бүтэц зэрэг маш олон янзын хувилбарыг уян хатан дэмжих шаардлагыг голчлон анхаарсан. Өмнө нь формат бүрт тусдаа парсер, код бичих шаардлагатай байсан нь засварлахад хүндрэлтэй, алдааны магадлал өндөр байв. Иймд дараах зарчмуудыг баримталсан:

- MappingConfigLoader нь буулгалт JSON-уудыг classpath-aac болон файл системээс ачаалах боломжтой. Ингэснээр deployment бүрт тохиргооны файлуудыг өөрчлөх, шинэ банк нэмэхэд кодонд өөрчлөлт хийхгүйгээр шийдэж чадна.
- Банк бүрийн буулгалтыг bankCode-оор индексжүүлж, хурдан lookup хийх боломжтой болгосон. Энэ нь олон буулгалт дундаас зөв буулгалтыг автоматаар сонгоход чухал.
- Файлын нэрээр буулгалтыг автоматаар олж сонгох логик нэмсэн. Ингэснээр хэрэглэгч буруу буулгалт сонгох эрсдэл багассан.
- Буулгалтуудыг runtime үед дахин ачаалах боломжтой болгосон. Энэ нь production орчинд шинэ буулгалт нэмэх, засварлахад системийг зогсоохгүйгээр шийдэж чадна.
- Буулгалтын файл буруу, дутуу, эсвэл parse алдаа гарсан тохиолдолд бусад буулгалтуудыг үргэлжлүүлэн ачаалж, системийн үндсэн үйл ажиллагаа тасалдахгүй байхаар шийдсэн.

Эдгээр шийдлүүдийн үр дүнд MinuPOS системд шинэ банк нэмэх, хуучин банкны формат өөрчлөгдөх, буулгалт-ийг засварлах зэрэг бүх үйлдлийг кодонд өөрчлөлт хийхгүйгээр зөвхөн буулгалт JSON-уудыг засах замаар шийддэг болсон. Энэ нь хөгжүүлэлтийн хурд, засварлахад хялбар байдал, алдааны магадлалыг эрс бууруулсан.

*6.3. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ХЭРЭГЖҮҮЛЭЛТ*

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

```
1 private void loadFromClasspath(String resourcePattern) {  
2     try {  
3         logger.info("Loading(mapping(resources(from(classpath(pattern  
4             '{}', resourcePattern);  
5             PathMatchingResourcePatternResolver resolver = new  
6                 PathMatchingResourcePatternResolver();  
7             Resource[] resources = resolver.getResources(resourcePattern);  
8             for (Resource r : resources) {  
9                 try {  
10                     JsonNode node = objectMapper.readTree(r.getInputStream  
11                         ());  
12                     addMapping(node);  
13                     logger.debug("Loaded(mapping(from(classpath(resource:{}  
14                         '{}", r.getFilename());  
15                 } catch (Exception e) {  
16                     logger.warn("Failed(to(parse(mapping(resource:{}:{}  
17                         r.getFilename(), e.getMessage());  
18                 }  
19             } catch (IOException e) {  
20                 logger.warn("No(mapping(resources(found(for(pattern:{}:{}  
21                         resourcePattern, e.getMessage());  
22             }  
23         }  
24     }
```

Код 6.10: Classpath-aac буулгалтын JSON-уудыг ачаалах

```
1 private void loadFromFilesystem(String folderPath) {  
2     File folder = new File(folderPath);
```

6.3. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ХЭРЭГЖҮҮЛЭЛТ

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

```
3     if (!folder.exists() || !folder.isDirectory()) {
4
5         logger.debug("Mappings\underline{folder} '{}' \u2014 does\underline{not} \u2014 exist \u2014 or \u2014 not \u2014 a\underline{directory}\u2014\u2014 skipping\underline{filesystem}\u2014load.", folderPath);
6
7     return;
8
9 }
10
11 logger.info("Loading\underline{mapping}\u2014JSON\u2014files\u2014from\u2014filesystem\u2014folder '{}'
12     ", folderPath);
13
14 File[] files = folder.listFiles((d, name) -> name.toLowerCase().
15
16     endsWith(".json"));
17
18 if (files == null) return;
19
20 Arrays.stream(files).sorted().forEach(f -> {
21
22     try {
23
24         String content = new String(Files.readAllBytes(f.toPath()))
25
26         ;
27
28         JsonNode node = objectMapper.readTree(content);
29
30         addMapping(node);
31
32         logger.debug("Loaded\underline{mapping}\u2014from\u2014filesystem\u2014file: \u2014{}",
33             f.
34
35             getName());
36
37     } catch (Exception e) {
38
39         logger.warn("Failed\u2014to\u2014parse\u2014mapping\u2014file: \u2014{}:\u2014{}",
40             f.
41
42             getName(), e.getMessage());
43
44     }
45
46 });
47
48 }
```

Код 6.11: Файл системээс mapping JSON-уудыг ачаалах

```
1 private void addMapping(JsonNode node) {
2
3     mappingsList.add(node);
```

```
4 JsonNode bankCodeNode = node.get("bankCode");

5 if (bankCodeNode != null && bankCodeNode.isTextual()) {

6     String bankCode = bankCodeNode.asText().trim();

7     if (!bankCode.isEmpty()) {

8         mappingsByBankCode.put(bankCode, node);

9     }

10 }

11 }
```

Код 6.12: Банкны кодоор mapping-ийг индексжүүлэх

```
17
18     } catch (Exception rex) {matching (strip .* common
19         pattern)
20
21         String plain = patternText.replace(".*", "");
22
23         if (!plain.isEmpty() && name.contains(plain)) {
24
25             return Optional.of(cfg);
26
27         }
28
29     }
30
31
32 // 2
33
34     if (sheetName != null && !sheetName.trim().isEmpty()) {
35
36         String sheet = sheetName.trim().toLowerCase();
37
38         for (JsonNode cfg : mappingsList) {
39
40             JsonNode sheetNode = cfg.get("sheetName");
41
42             if (sheetNode != null && sheetNode.isTextual()) {
43
44                 String s = sheetNode.asText().trim().toLowerCase();
45
46                 if (!s.isEmpty() && sheet.contains(s)) {
47
48                     return Optional.of(cfg);
49
50                 }
51
52             }
53
54         }
55
56     }
57
58 // 3
59
60     if (mappingsList.size() == 1) {
61
62         return Optional.of(mappingsList.get(0));
63
64     }
65
66 }
```

```
44     }
45     return Optional.empty();
46 }
```

Код 6.13: Файлын нэрээр mapping автоматаар сонгох

```
1 public synchronized void reload() {  
2     try {  
3         loadMappings();  
4     } catch (IOException e) {  
5         logger.error("Failed to reload mapping files: {}", e.getMessage(),  
6                     e);  
7     }  
}
```

Код 6.14: Runtime reload хийх capability

7. ДҮГНЭЛТ

7.1 Yр дүн

	statement_id	txn_amount	txn_date	txn_desc
1	UNKNOWN-uploaded-17608877245504346231-Statement_MNT_5564190560.xls	[null]	[null]	Гүйлгээний утга
2	UNKNOWN-uploaded-17608877245504346231-Statement_MNT_5564190560.xls	14000.00	2025-08-15 09:02:29	qpay 200000615958779 MBA_P175521973269781240 1 PRE
3	UNKNOWN-uploaded-17608877245504346231-Statement_MNT_5564190560.xls	50.00	2025-08-15 09:02:34	Ухаалаг мэдээ ўйлчилгээний хураамж
4	UNKNOWN-uploaded-17608877245504346231-Statement_MNT_5564190560.xls	-14050.00	[null]	[null]
5	UNKNOWN-uploaded-1266222939347116961-Statement_MNT_5564190560.xls	[null]	[null]	Гүйлгээний утга
6	UNKNOWN-uploaded-1266222939347116961-Statement_MNT_5564190560.xls	36000.00	2025-08-17 15:01:54	TRF=002022378602-949628XXXXXX0336-MONOS SELBE>Baya
7	UNKNOWN-uploaded-1266222939347116961-Statement_MNT_5564190560.xls	50.00	2025-08-17 15:01:59	Ухаалаг мэдээ ўйлчилгээний хураамж
8	UNKNOWN-uploaded-1266222939347116961-Statement_MNT_5564190560.xls	-36050.00	[null]	[null]

Total rows: 8 of 8 Query complete 00:00:00.094 Ln 2, Col 1

Зураг 7.1: Гүйлгээний мэдээллүүдийг өгөгдлийн санд амжилттай оруулсан байдал

7.2 Yр дүнгийн тайлан

Монгол Ай Ди компанид 21 хоногийн хугацаанд үйлдвэрлэлийн дадлагыг хийснээр би онолын мэдлэгээ практиктай холбон, программ хангамжийн хөгжүүлэлтийн бодит орчинд туршлага хуриатлуулав. Энэ хугацаанд би байгууллагын технологийн багтай хамтран ажиллаж, баг хоорондын зохион байгуулалт, Agile зэрэг хөгжүүлэлтийн орчин үеийн арга зүйтэй танилцсан. Дадлагын удирдагч, Технологи хариуцсан захирал Х. Ганзориг надад Java технологид суурилсан нэг модулийг сайжруулах ажлыг даалгасан. Би байгууллагын техникийн баримт бичгүүдийг судалж, өөрийн санааг нэмэн тухайн модулийг амжилттай сайжруулж, ашиглалтад өгөв. Энэ төсөл нь миний шийдвэр гаргах, асуудал шийдвэрлэх чадварыг дээшлүүлсэн юм. Дадлагын эхэнд тавьсан программ хангамжийн загварууд болон хөгжүүлэлтийн арга барилуудыг судлах зорилгodoо хүрснээс гадна, өөрийн мэргэжлийн ирээдүйн чиг хандлагыг тодорхойлж, үүнд хүрэхэд шаардлагатай хувийн хөгжлийн төлөвлөгөөг гаргах боломжтой болсон. Энэхүү дадлага нь миний хувьд мэдлэг, ур чадвараа нэгтгэн, ирээдүйн карьертаа бэлтгэх чухал алхам боллоо.

Bibliography

- [1] Нарангэрэл, Н., Болд, Л., Өнөрбаян, Ц. ба бусад. *Монгол хэлний зөв бичих дурмийн журамласан толь*. Улаанбаатар хот, 2022. URL: <https://toli.gov.mn/>
- [2] Чулуунбанди, Н., Лхагвасурэн, Т., Дугарчuluun, Г. ба бусад. *Мэдээлэл, харилцаа холбооны технологийн англи нэр томоёоны тайлбар*. Линограф, 2018.
- [3] Refactoring Guru. *Refactoring and Design Patterns*. 2014-2024. URL: <https://refactoring.guru/>
- [4] Bass, L., Clements, P., & Kazman, R. *Software architecture in practice*. 3rd ed. Addison-Wesley, 2012.
- [5] Mittelbach, F., Goossens, M., Braams, J., Carlisle, D., & Rowley, C. *The LaTeX Companion*. 2nd ed. Addison-Wesley, 2004.
- [6] Knuth, D. E. *The TeXbook*. Addison-Wesley, 1984.
- [7] Lamport, L. *LaTeX: A Document Preparation System*. 2nd ed. Addison-Wesley, 1994.
- [8] W3Schools. *HTML Tutorial*. URL: <https://www.w3schools.com/html/>
- [9] Python Software Foundation. *Python Language Reference*, version 3.11. URL: <https://www.python.org/>
- [10] GitHub. *GitHub Documentation*. URL: <https://docs.github.com/>

А. НЭРШЛИЙН КОНВЕНЦ

Table A.1: Auftragsverwaltung систем дээрх Герман-Англи нэршлийн конвенц

№	Герман	Англи
1	auftragsverwaltung	order management
2	artikel	item
3	kunde	customer
4	benutzer	user
5	geld	money
6	betrag	amount
7	anzNachKomma	number after decimal point
8	waehrung	currency
9	addieren	add
10	subtrahieren	subtract
11	multiplizieren	multiply
12	umrechnen	convert
13	kompatibel	compatible
14	betragMitKomma	amount with decimal point
15	artikelnr	Item No.
16	artikelbezeichnung	Item Description
17	pries	Price
18	mindestbestand	Minimum Stock
19	bestand	Stock
20	aktualisieren	Update
21	lieferdatum	delivery date
22	apositionen	positions
23	einfuegen	insert
24	entfernen	remove
25	loeschen	delete
26	anrede	Title
27	vorname	First name
28	strasse	Street
29	plz	Zip code
30	ort	City
31	debitorennr	Account receivable number
32	eine artikel verwaltung	an article management
33	artikelliste	article list
34	datenquelle	data source
35	hinzufuegen	add
36	entfernen	remove
37	finden	find
38	findeArtikelMitUnterdeckung	find article with shortfall
39	posNr	PosNo

(үргэлжилнэ)

Table A.1 – Үргэлжлэл

№	Герман	Англи
40	menge	Quantity
41	tatsLieferdatum	Actual Delivery Date
42	istAbgeschlossen	isCompleted
43	findeAuftraegeZuArtikel	Find orders by item
44	findeAuftraegeZuKunde	Find orders by customer
45	naechsteAuftragsnr	Next order number
46	kennwortHash	password hash
47	fabrik	factory
48	verbindung	connection
49	meinlocale	myLocale
50	berechtigungssystem	authorization system

В. БАНКНЫ ХУУЛГЫН ФОРМАТ

The screenshot shows an Excel spreadsheet titled "Statement_MNT_5564190560". The spreadsheet contains the following data:

ХААН БАНК						Printed Date:	2025-08-15
Депозит дансны дэлгэрэнгүй хуулга						Интервал: 2025-08-15-2025-08-15	
Харяглагч:			ЦЭДЭН-ИШ БИНДЭРЦЭЦЭГ				
Дансны дугаар:			5564190560				
Гүйлгээний огноо	Салбар	Эхний үлдэгдэл	Дебит гүйлгээ	Кредит гүйлгээ	Эцсийн үлдэгдэл	Гүйлгээний утга	Харьцаан данс
2025-08-15 09:02:29	5000	79567.28	14000.00	0	65567.28	прау 200000615958779 MBA_P175521973269781240 1 PRE	5038085431
2025-08-15 09:02:34	5564	65567.28	50.00	0	65517.28	Ухаалаг мэдээ ўйлчилгээний хурамж	
Нийт дун: 14050.00 0							

Зураг В.1: Хаан банкны гүйлгээний Excel хуулгын формат

C. БУУЛГАЛТЫН ЗАГВАР

```
1  {
2      "bankCode": "KHANBANK",
3      "sheet": 0,
4      "headerRowIndex": 6,
5      "mapping": {
6          "statement": {
7              "STATEMENT_ID": {"colIndex": "-1"},
8              "ACCOUNT_NO": {"colIndex": ""}}
9      },
10     "detail": {
11         "DETAIL_ID": {"colIndex": "-1"},  

12         "STATEMENT_ID": {"colIndex": "-1"},  

13         "TXN_DATE": {"colIndex": "0", "type": "date"},  

14         "PRE_BALANCE": {"colIndex": "2", "type": "number"},  

15         "DEBIT": {"colIndex": "3", "type": "number"},  

16         "CREDIT": {"colIndex": "4", "type": "number"},  

17         "POST_BALANCE": {"colIndex": "5", "type": "number"},  

18         "TXN_DESC": {"colIndex": "6"},  

19         "CO_ACCOUNT": {"colIndex": "7"}}
20     }
21 }
22 }
```

Код C.1: Хаан банкны гүйлгээний Excel хуулгын форматын буулгалтын загвар