

**МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ**

Цэдэн-Ишийн Биндэрцэцэг

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

Програм Хангамж (D061302)
Үйлдвэрлэлийн дадлагын тайлан

Улаанбаатар хот

2025 оны 9 сар

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

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

Програм 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 ”Програм 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.
.....төлөвлөгөөнийжилүүдэшигүүрчүүлэхжилийн
.....хэрэгжүүлэх сал.

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

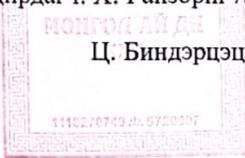
ДАДЛАГЫН АЖЛЫН ТӨЛӨВЛӨГӨӨ

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 Програм хангамжийн зохиомжийн зарчмууд	4
3.2 Програм хангамжийн зохиомжийн үлгэр загварууд: Байгуулалтын, Бүтцийн, Зан байдлын	5
4. АШИГЛАСАН ТЕХНОЛОГИ	13
4.1 Git	13
4.2 React library	13
4.3 Next.js	15
4.4 Material-ui сан	16
5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ	18
5.1 Урвуу инженерчлэл: "Auftragsverwaltung" систем дэх зохиомжийн үлгэр загварууд	18
5.2 МУИС-ийн дипломын ажлыг удирдах системийн зохиомж дахь үлгэр загваруудын хэрэглээ	28
6. ДАДЛАГЫН ЯВЦ	30

6.1 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн шинжилгээ	30
6.2 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн зохиомж	40
6.3 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн хэрэгжүүлэлт	44
7. ДҮГНЭЛТ	46
7.1 Yp дүн	46
7.2 Yp дүнгийн тайлан	49
НОМ ЗҮЙ	50
ХАВСРАЛТ	51
A. КОНВЕНЦ	51
B. КОДЫН ХЭРЭГЖҮҮЛЭЛТ	53
B.1 Форм	53
B.2 Toast компонент	58
B.3 Toast компонент дээр интерфэйсийн автоматжуулсан тест бичсэн байдал	63

ЗУРГИЙН ЖАГСААЛТ

5.1	”Applicationlogic” багцын классын диаграм	20
5.2	”Utility” багцын классын диаграм	21
5.3	”DAO” багцын классын диаграм	25
5.4	Дипломын ажлыг удирдах системийн классын диаграм	29
6.1	MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн ажлын явцын диаграм	32
6.2	Excel хуулга боловсруулах ерөнхий төлөвийн диаграм	40
6.3	”Builder” үлгэр загварын зохиомжийн диаграм	42
6.4	MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн model багцын классын диаграм	42
6.5	MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн зохиомжийн үеийн классын диаграм	43
7.1	Формын эхний хувилбар	46
7.2	Material ui болон React Select ашигласан байдал	47
7.3	Хүсэлт амжилттай болсон үед харагдах Toast	48
7.4	Хүсэлт амжилтгүй болсон үед харагдах Toast	48

ХҮСНЭГТИЙН ЖАГСААЛТ

6.1	Ажлын явц: Банкны хуулга оруулах (UC-FS-01)	33
6.2	Ажлын явц: Банкны загвар үүсгэх (UC-FS-02)	34
6.3	Банкны Excel хуулгыг боловсруулах модулийн функциональ шаардлага	36
6.4	Банкны Excel хуулгыг боловсруулах модулийн функциональ бус шаардлага	37
6.5	Банкны Excel хуулгыг боловсруулах модулийн хөгжүүлэлтийн орчинд тавигдах шаардлага	37
6.6	MinuPOS системийн F стандарт бүтцийн толгой баганууд	38
6.7	MinuPOS системийн F стандарт бүтцийн дэлгэрэнгүй баганууд	39
A.1	Auftragsverwaltung систем дээрх Герман-Англи нэршлийн конвенц	51

Кодын жагсаалт

4.1	JSX ашиглаж ”container” кластай html элемент буцаах компонент	14
4.2	Material-ui сангийн компонентыг ашиглаж буй байдал	16
5.1	Order классын устгагч auftragLoeschen	19
5.2	Order классын арга getAuftragssumme	22
5.3	OrderManagement классын арга hinzufuegen	23
5.4	KundeTO класс	25

УДИРТГАЛ

Миний бие Цэдэн-Ишийн Биндэрцэцэг нь үйлдвэрлэлийн дадлагын З долоо хоногийн хугацаанд програм 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 програм 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:

- Програм 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 модульийг зохиомжлоно;
- Spring Boot фреймворк ашиглан MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийг хэрэгжүүлнэ.

3. ОНОЛЫН СУДАЛГАА

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

3.1 Програм хангамжийн зохиомжийн зарчмууд

Програм хангамжийн зохиомжийн зарчмууд нь сайн зохиомжийг бий болгоход чиглэсэн удирдамж, хамгийн сайн туршлагуудын цуглуулга юм. Эдгээр зарчмууд нь програм хангамжийн системийг уян хатан, дахин ашиглах боломжтой, засвар үйлчилгээ хийхэд хялбар болгоход тусалдаг. 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:** Програм 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 интерфэйсүүдэд хуваах ёстой. Ингэснээр клиент хэрэгцээгүй аргуудыг хэрэгжүүлэх шаардлагагүй болно.

3.2. ПРОГРАМ 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.

3.2 Програм хангамжийн зохиомжийн үлгэр загварууд:

Байгуулалтын, Бүтцийн, Зан байдлын

Зохиомжийн үлгэр загвар нь програм хангамжийн зохиомжийн нийтлэг асуудлуудад батлагдсан, 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 програмчлалын 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. ПРОГРАМ 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.

”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 програмыг эхлүүлэх үед биш, харин анх удаа хэрэг болох хүртэл хойшлуулдаг оновчлолын арга зам бөгөөд хэзээ ч ашиглагдахгүй объект байгуулахаас зайлсхийж, програмыг эхлүүлэх хугацааг багасгаж, санах ойн ашиглалтыг бууруулдаг.

3.2. ПРОГРАМ 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.

”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. ПРОГРАМ ХАНГАМЖИЙН ЗОХИОМЖИЙН ҮЛГЭР ЗАГВАРУУД: БАЙГУУЛАЛТЫН, БҮТЦИЙН, ЗАН БАЙДЛЫН БҮЛЭГ 3. ОНОЛЫН СУДАЛГАА

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

Аппликешныг олон төрлийн бүтээгдэхүүний аль нэгэнд тохируулах 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. Конкreet хийсвэр үйлдвэрийн класс нь холбогдох бүтээгдэхүүн бүрийг үйлдвэрлэхийн тулд олон үйлдвэрийн аргын дуудлага ашигладаг.

”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 объектуутдтай 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. ПРОГРАМ 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 програмын 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.

”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 програм ажиллаж байх

3.2. ПРОГРАМ 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.

3.2.2 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. ПРОГРАМ 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.

”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.

Програм 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. ПРОГРАМ 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 дибаг хийхэд хэцүү байж болно.

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.

4. АШИГЛАСАН ТЕХНОЛОГИ

4.1 Git

Linus Torvalds буюу Linux Kernel-г хөгжүүлсэн хүн Kernel-ийнхээ эх кодыг удирдах зорилгоор уг технологийг анх санаачилж, хэрэгжүүлсэн байдаг. Гол зорилго нь Version Control System буюу хувилбар удирдах системийг бүтээх ба ингэснээр хөгжүүлэлтийн явцад бүх өөрчлөлтөө хадгалах, багаар ажиллах боломжийг бүрдүүлж өгсөн юм. Дадлага хийх хугацаандаа байгууллагынхаа сонгосон Gitlab.com платформыг ашиглаж, GIT-н үндсэн үйл ажиллагааны талаар илүү сайн ойлголттой боллоо.

Үндсэн ажиллагааг нь тайлбарлавал GIT ашиглаж буй хөгжүүлэгч/хэрэглэгч бүр өөрийн төхөөрөмж дээр үндсэн repository-тай яг ижил repository-г үүсгэнэ. Ингэснээр сүлжээнд холбогдсон эсэхээс үл хамааран дурын өөрчлөлт, хөгжүүлэлтээ хийх боломжтой ба уг ажил дууссан үед бичсэн өөрчлөлтүүдээ commit хийж, remote repository руу push үйлдлийг хийнэ.

4.2 React library

Фэйсбүүк компани дотооддоо ашиглаж байсан технологио 2013 онд танилцуулсан нь програмчлалын Javascript хэлийг ашиглаж хийсэн Front-end library болох React¹ технологи юм. Declarative UI хөгжүүлэлтийн аргыг хамгийн анх дэлгэрүүлж, өргөн хэрэглээнд нэвтрүүлж чадсан тул Declarative UI-н гол төлөөлөгч гэж явдаг. Уг технологийг ашиглахын тулд үндсэн хэдэн ойлголтууд авах хэрэгтэй. Үүнд component ба түүний lifecycle, javascript-н өргөжүүлсэн хувилбар болох jsx, мөн хамгийн чухал зүйл болох Virtual DOM нар багтана.

Declarative UI гэдэг нь хэрэглэгчийн интерфэйсийн кодыг бичихдээ юу зурагдах буюу render хийх үеийн интерфэйсийг бүгдийг урьдчилан тодорхойлдог. Imperative програмчлалаас ялгаатай нь хязгаартай нөхцөлд яг юу хийхийг хатуугаар зааж өгөхгүйгээр тухайн state-с

¹Reactjs official site <https://reactjs.org>

хамааруулж хэрэглэгчийн хүссэн зүйлийг гаргаж өгөх боломжтой.

React нь component-based буюу DOM дээр хэвлэж байгаа бүх зүйлс component байна гэсэн дүрмийг баримталдаг. Component үүсгэж бичихийн давуу тал нь нэг бичсэн кодоо олон дахин бичигдэхээс зайлсхийж, дахин ашиглах боломжийг олгодог. Тус бур өөрсдийн гэсэн дотоод төлөвтэй мөн гаднаас утга хүлээн авах чадвартай. Үүнийг бид Props гэж нэрлэдэг. Мөн component нь stateless, stateful гэж хоёр хуваагддаг ба stateful component нь өөрийн гэсэн төлөвтэй, түүнийгээ удирддаг, class болон hook ашигласан функцууд байна. React-н давуу тал нь state эсвэл props-н өөрчлөлтийг үргэлж хянаж байдаг тул өөрчлөлт орж ирэхэд бүтэн хуудсыг зурах бус зөвхөн тухайн өөрчлөгдсөн component-г л дахин зурдаг. Ингэснээр энгийн вэбүүдээс илүү хурдтай ажилладаг.

JSX нь Javascript Extended гэсэн үгний товчлол бөгөөд энгийнээр javascript дотор HTML-н тагуудыг бичиж өгөх мөн кодыг илүү богино болгож хүссэн үр дүндээ хүрэх боломжийг олгодог. Үүний цаана Babel гэсэн transcompiler-г ашиглаж дундын хөрвүүлэлтийг хийдэг ба хэдийгээр HTML таг бичиж байгаа харагддаг ч код дунд цэвэр HTML-г огтоос бичиж өгдөггүй гэсэн үг юм.

```

1  export function Container = ({children}) => {
2
3      return (
4          <div className="container">
5              {children}
6          </div>
7      )
8
9  }
```

Код 4.1: JSX ашиглаж ”container” класстай html элемент буцаах компонент

Жинхэнэ DOM дээр богино хугацаанд олон өөрчлөлт хийхэд удах асуудал гарсан тул React маань Virtual DOM гэсэн abstraction давхарга үүсгэж өөрчлөлтүүдээ Virtual DOM дээрээ хадгалаад нэгдсэн нэг өөрчлөлтийг жинхэнэ DOM руугаа дамжуулдаг.

4.3 Next.js

Next.js² нь React library дээр суурилж хөгжүүлсэн нээлттэй эхийн фрэймворк бөгөөд Vercel компани 2016 онд албан ёсны танилцуулгаа хийж олон нийтэд зарласан юм. React нь зөвхөн хэрэглэгчийн интерфэйсийг зурах үүрэгтэй сан ба бусад вэб хөгжүүлэлтэд хэрэгтэй хуудас хооронд шилжих гэх мэт үйлдлийг react-router болон бусад маш олон нэмэлт сангаас сонголт хийж шийдэх шаардлагатай байсан нь төслийн эхлэх явцыг удаашруулах хандлагатай байдаг. Харин Next.js ашигласнаар нэг ч тохиргоо хийлгүйгээр төслийг эхлүүлж шууд код бичих боломжийг бүрдүүлдэг. Цаана нь хийгдсэн тохиргоо нь нийт вэбсайтуудын 90 хувийн шаардлагыг хангаж чаддаг гэж үздэг нь уг фрэймворкын сүүлийн жилүүдэд эрэлттэй болж буй шалтгаануудыг нэг билээ. Дадлага хийсэн компани маань төслүүдээ Next.js дээр хийдэг тул уг технологийг зайлшгүй сурх шаардлага гарсан юм.

Next.js давуу талуудаас дурьдвал:

- Image Optimization буюу их хэмжээтэй зураг оруулахад автоматаар зургийн чанарыг алдагдалгүйгээр хэмжээг багасгаж өгдөг
- Zero config буюу нэг ч тохиргоо хийлгүйгээр төслөө эхлүүлэх боломж
- Static Site Generator болон Server Side Render хийх
- Typescript болон Fast Refresh дэмждэг
- File-system Routing буюу “pages” гэсэн хавтас дотор үүссэн файлуудаас хамаарч вэбийн замууд тодорхойлогддог мөн dynamic routing ашиглах боломжтой
- API Routes буюу өөр дээрээ nodejs сервер ашиглаж API endpoint гаргах. Ингэснээр тусдаа сервер ашиглах шаардлага үүсэхгүй
- SEO буюу хайлтын системийн оновчлолыг SSR ашиглаж тохируулж өгөх гэх мэт маш олон давуу талуудтай

²Next.js official site <https://nextjs.org>

Мөн ердөө ганц “build” командаар статик болон динамик вэбийг гарган авч ямар нэгэн вэб сервер /apache, nginx гэх мэт/ ашиглалгүйгээр сервер дээрээ шууд байршуулах боломжтой юм.

4.4 Material-ui сан

Material-ui нь 2014 онд Google-н дизайнер Матиас Дуартегийн санаачилсан Material Design гэх design language дээр суурилж хийгдсэн нээлттэй эхийн төсөл юм. Ашигласан технологи нь React тул бэлэн component-уудыг дуудаж төсөл дээрээ шууд ашиглах боломжтой. Уг санг дадлагын хугацаан дахь сүүлийн долоо хоног дээрээ ажилласан төсөл дээр хэрэглэсэн ба дизайн дээр нэмэлт хөгжүүлэлт хийлгүй хариуцаж авсан component-ийнхoo зөвхөн логик үйлдлүүд дээр анхаарах боломжийг олгосноор хөгжүүлэлтийн процесийг маш ихээр хурдастгаж өгсөн.

Уг сан нь css-in-js технологи болох JSS-г ашигладаг. Хэрэв интерфэйс загвар дээр нэмэлт хөгжүүлэлт хийх шаардлага гарвал makeStyles() гэсэн custom hook ашиглаж CSS кодоо бичнэ. Сүүлийн beta хувилбар дээрээ JSS-г хасаж dynamic style дээрээ emotion болон styled-component-г хэрэглэж эхэлсэн байгаа. Жишээ болгон нэг компонентийн загварыг сольж үзүүллээ.

```

1 import {Grid, Card} from '@material-ui/core'

2

3 function Card({ children }) {
4
5   return (
6     <Grid item xs={12} md={4}>
7
8       <Card header="  □ ?">
9         {children}
10
11       </Card>
12     </Grid>
13   );
14 }

```

Код 4.2: Material-ui сангийн компонентыг ашиглаж буй байдал

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 програм 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 програм 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>

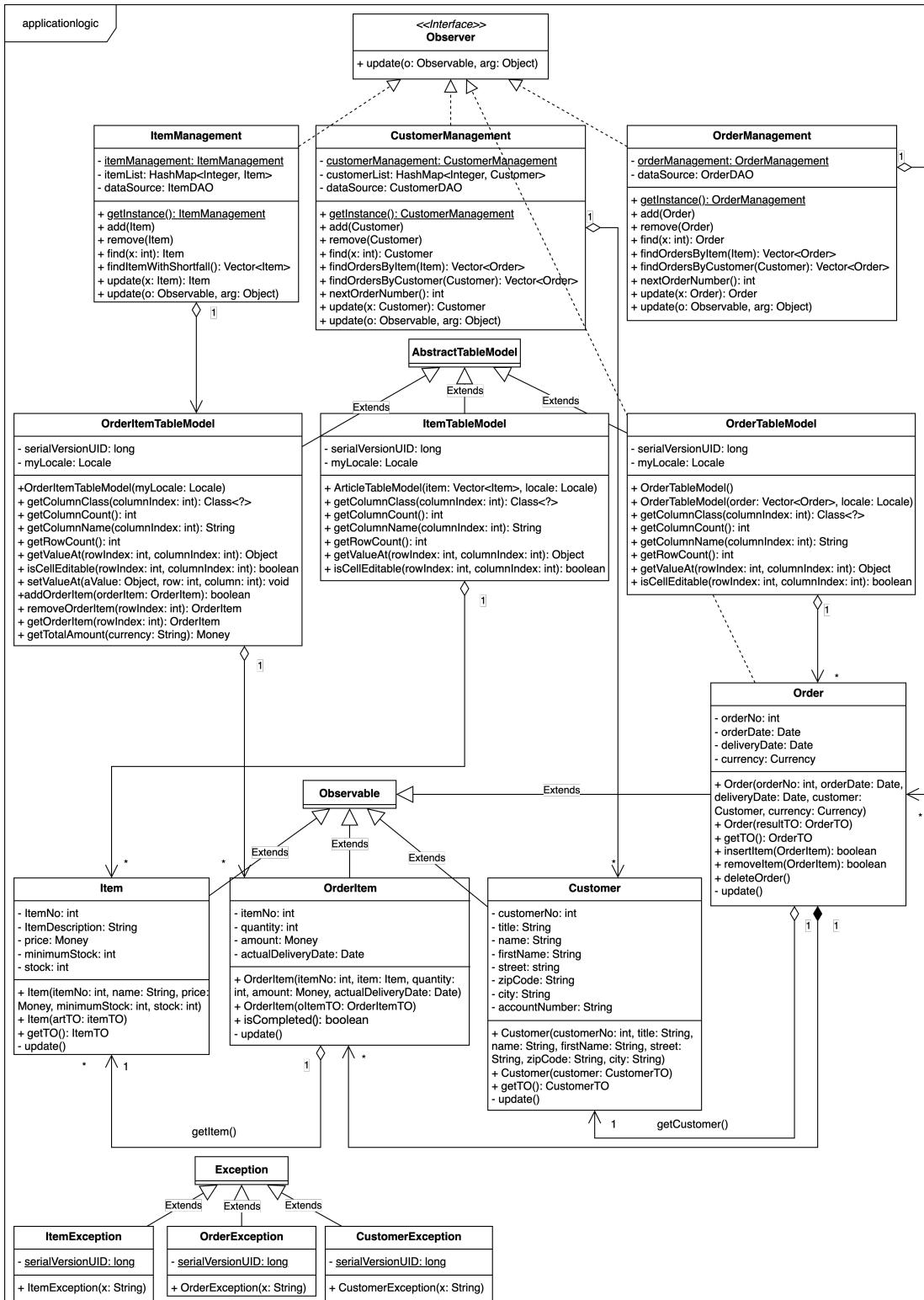
5.1. УРВУУ ИНЖЕНЕРЧЛЭЛ: "AUFTRAGSVERWALTUNG" СИСТЕМДЭХ
ЗОХИОМЖИЙН УЛГЭР ЗАГВАРУУД БҮЛЭГ 5. ДАДЛАГЫН БЭЛТГЭЛ АЖИЛ

```
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 }
```

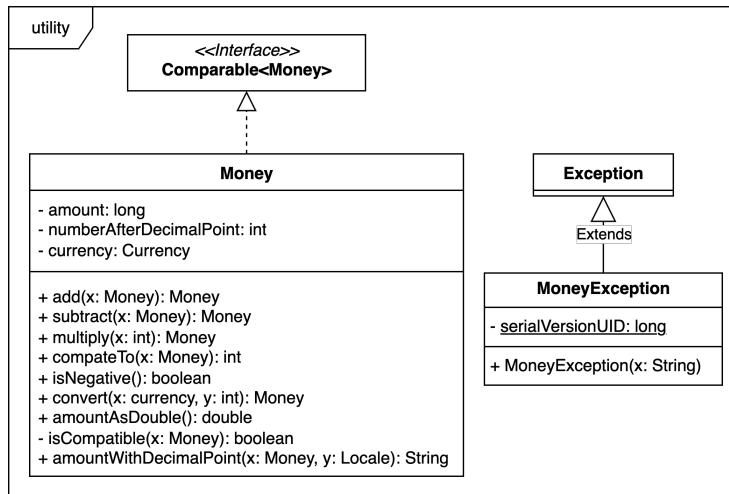
Код 5.1: Order классын устгагч auftragLoeschen

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 програмчлалд 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 програмчлалын 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

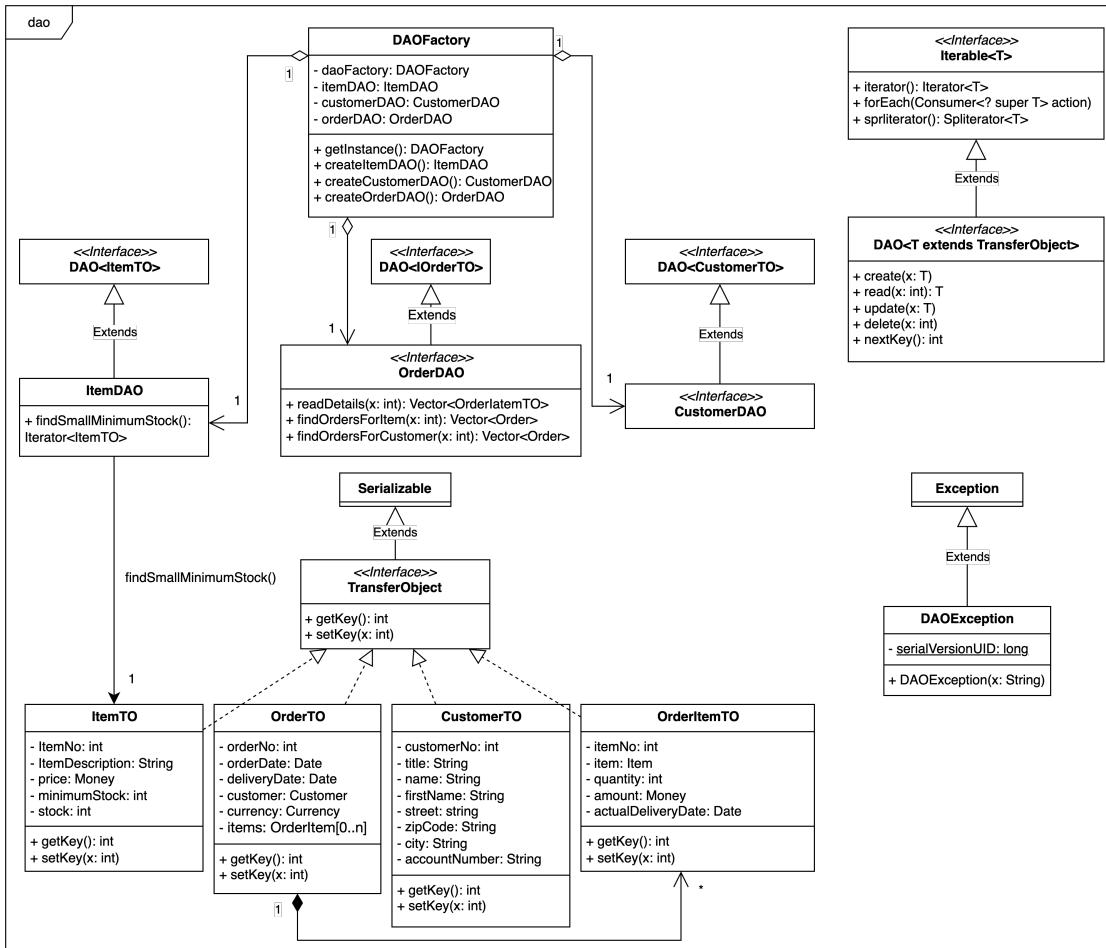
OrderManagement класын `hinzufuegen()` арга дотор `auf.addObserver(this)` гэж байгаа нь Singleton объект өөрөө Observable объектын өөрчлөлтийг хүлээн авч DAO⁴-руу өөрчлөлт илгээж байна. Үүнийг ажигласнаар "Observer" үлгэр загвар ашиглагдсан гэж дүгнэлээ. Домайн объект болох Customer, Order, OrderItem, 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 объектоос ирсэн өөрчлөлтийг хүлээн авч байна. Ингэснээр системийн бүх чухал домэйн объектууд үзэгдлийн төв болж, менежер бүр эдгээр үзэгдлийг сонсож, өөрийн цуглуулгад харгалзах домэйн объектыг дахин оруулах/шинэчлэх нь төвлөрсөн удирдлага үүсгэхээс гадна домэйн объект, 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 объектын цуглуулгыг удирдах үүрэгтэй болохыг илтгэнэ. Жишээ нь, доорх кодонд CustomerDAO нь Customer домэйн объектын цуглуулгыг удирдаж байна. Үүнийг ажигласнаар "Composite" үлгэр загвар ашиглагдсан гэж дүгнэлээ. Учир нь DAO интерфэйсийг хэрэгжүүлсэн класс бүр нь өөрийн төрөлд харгалзах домэйн объектын цуглуулгыг удирдах үүрэгтэй бөгөөд эдгээр DAO-уудын цуглуулга нь системийн бүх домэйн объектын 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 програмын 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

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

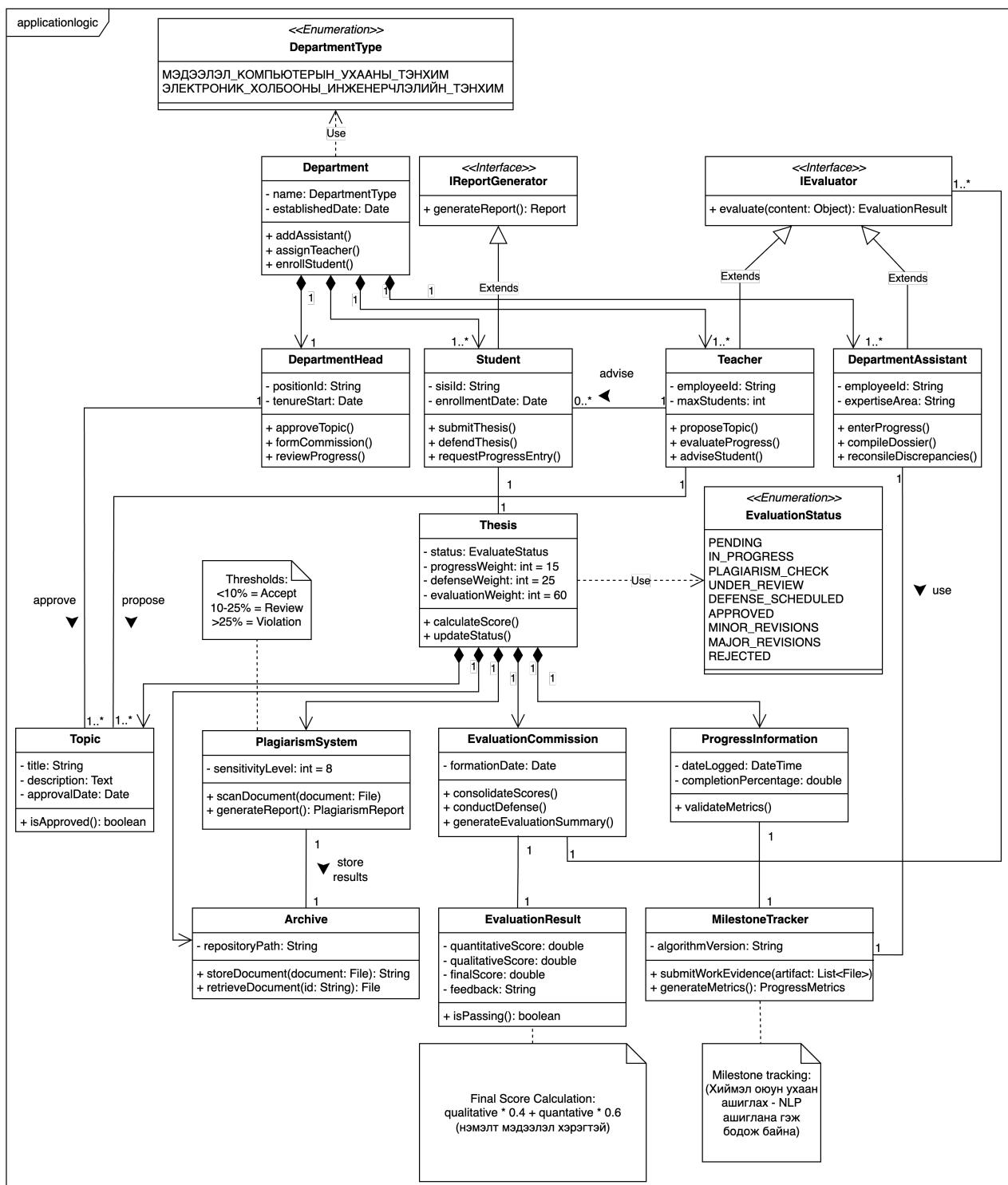
ҮЛГЭР ЗАГВАРУУДЫН ХЭРЭГЛЭЭ

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

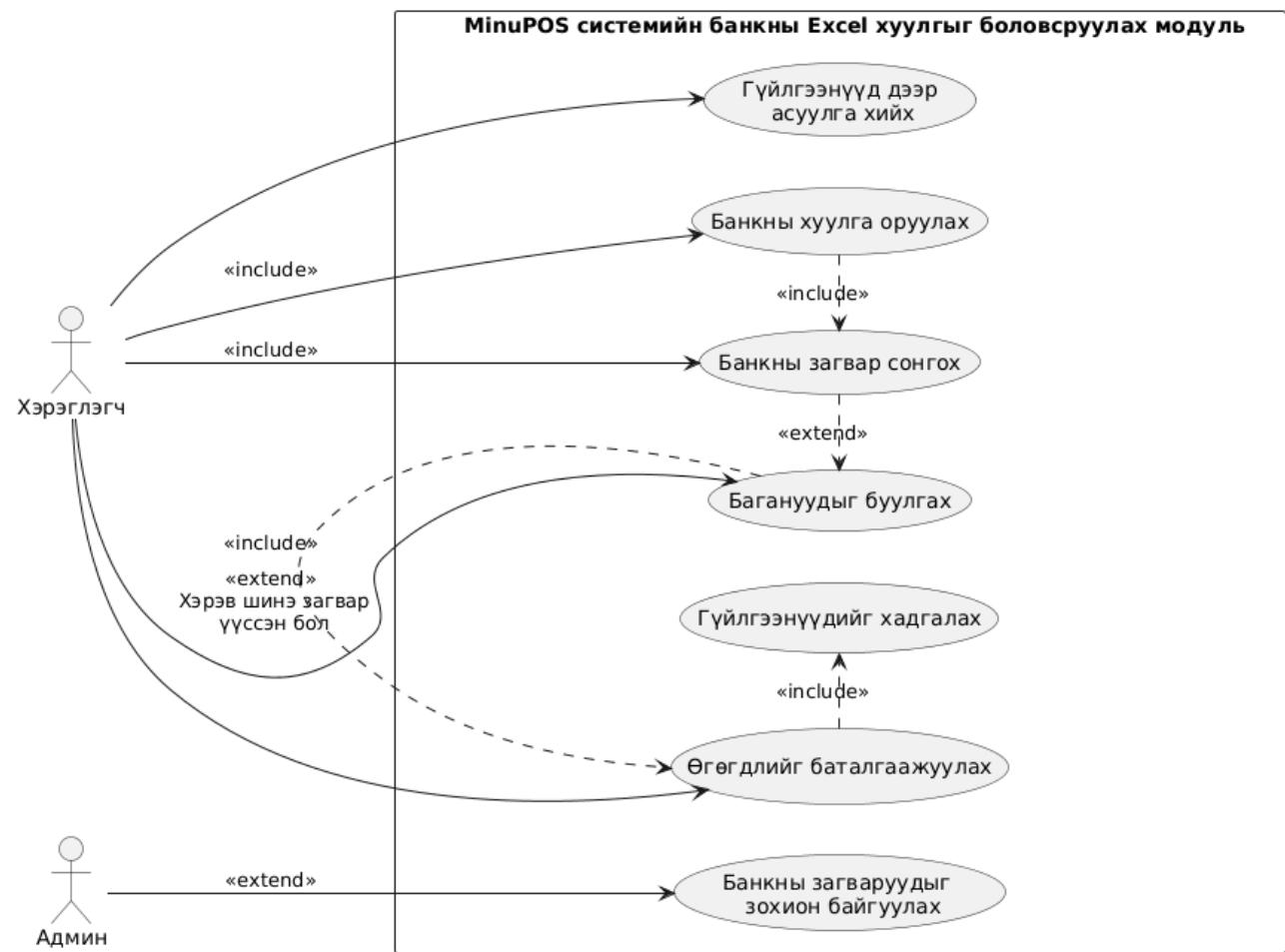
MinuPOS системийн системийн сервер талын код нь Жава технологи дээр бичигдсэн ба үндсэн фреймворк нь Spring Boot юм. MinuPOS нь SoftPOS шийдэлтэй; ухаалаг утсаар карт/NFC төлбөр хүлээн авах боломжийг MineSec SoftPOS¹ -оор хангадаг. MinuPOS системийн худалдан авагч/борлуулагч талын интерфэйс нь POS терминал, 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

6.1.4 Динамик загвар

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



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

Дээрх ажлын явцын диаграмд тусгасан чухал ажлын явцын тайлбаруудыг харгалзах 6.1 болон 6.2 хүснэгтүүдэд үзүүлэв. Энд ажлын явцын өдөөгч 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)

6.1.5 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

6.2. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ МОДУЛИЙН ЗОХИОМЖ

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ

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 "Factory" үлгэр загварын зохиомжийн тайлбар

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

6.2.4 "Builder" үлгэр загварын зохиомжийн тайлбар

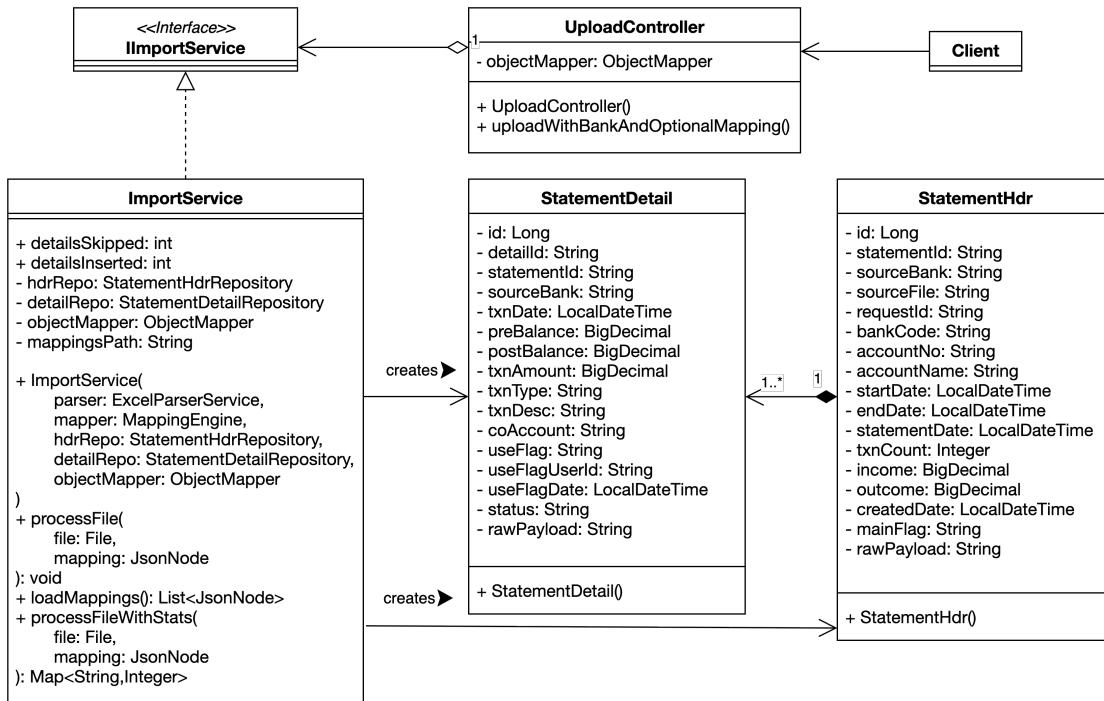
Модуль тодорх StatementHdr, StatementDetail зэрэг энтити обьектуудыг үүсгэхэд "Builder" үлгэр загварыг ашигласан (6.3 зургийг харна уу). Эдгээр энтитинүүд нь олон талбартай бөгөөд бүх утгууд нь нэг дор бэлэн байдаггүй. Барилгачин нь урт байгуулагч бичихгүйгээр, эсвэл бүх талбарыг гараар тохируулахгүйгээр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

6.2.5 Модулийн статик загвар

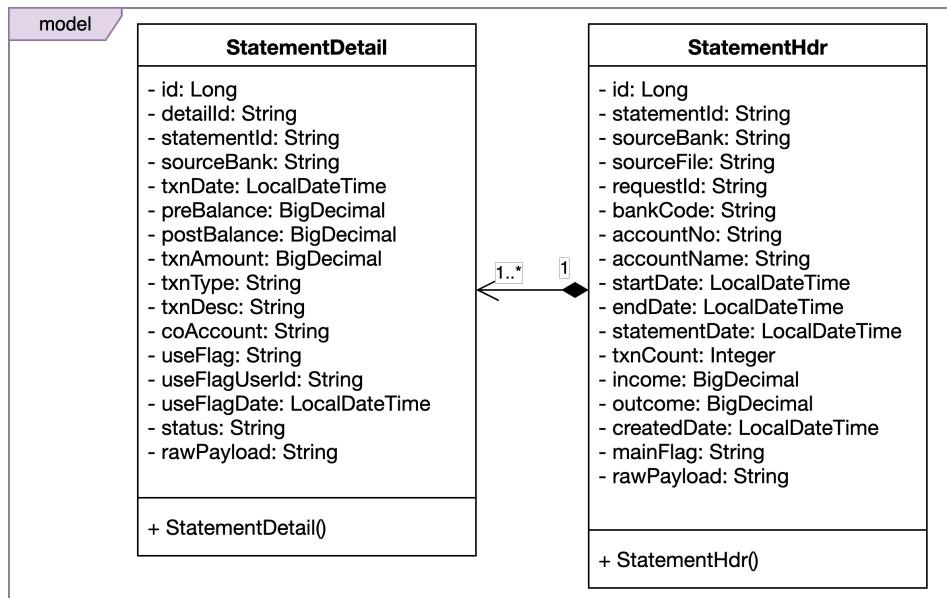
Шинжилгээний үед тодорхойлсон өгөгдлийн F стандарт бүтэцтэй нийцэхээр model багцын классын диаграмыг 6.4 зурагт үзүүлэв. Мэдээж getter, setter, toString зэрэг агуудыг агуулах бөгөөд эдгээрийг диаграмд оруулаагүй болно. Модулийн гол зохиомжийн диаграмыг 6.5 зурагт үзүүлэв.

**6.2. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ
МОДУЛИЙН ЗОХИОМЖ**

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ



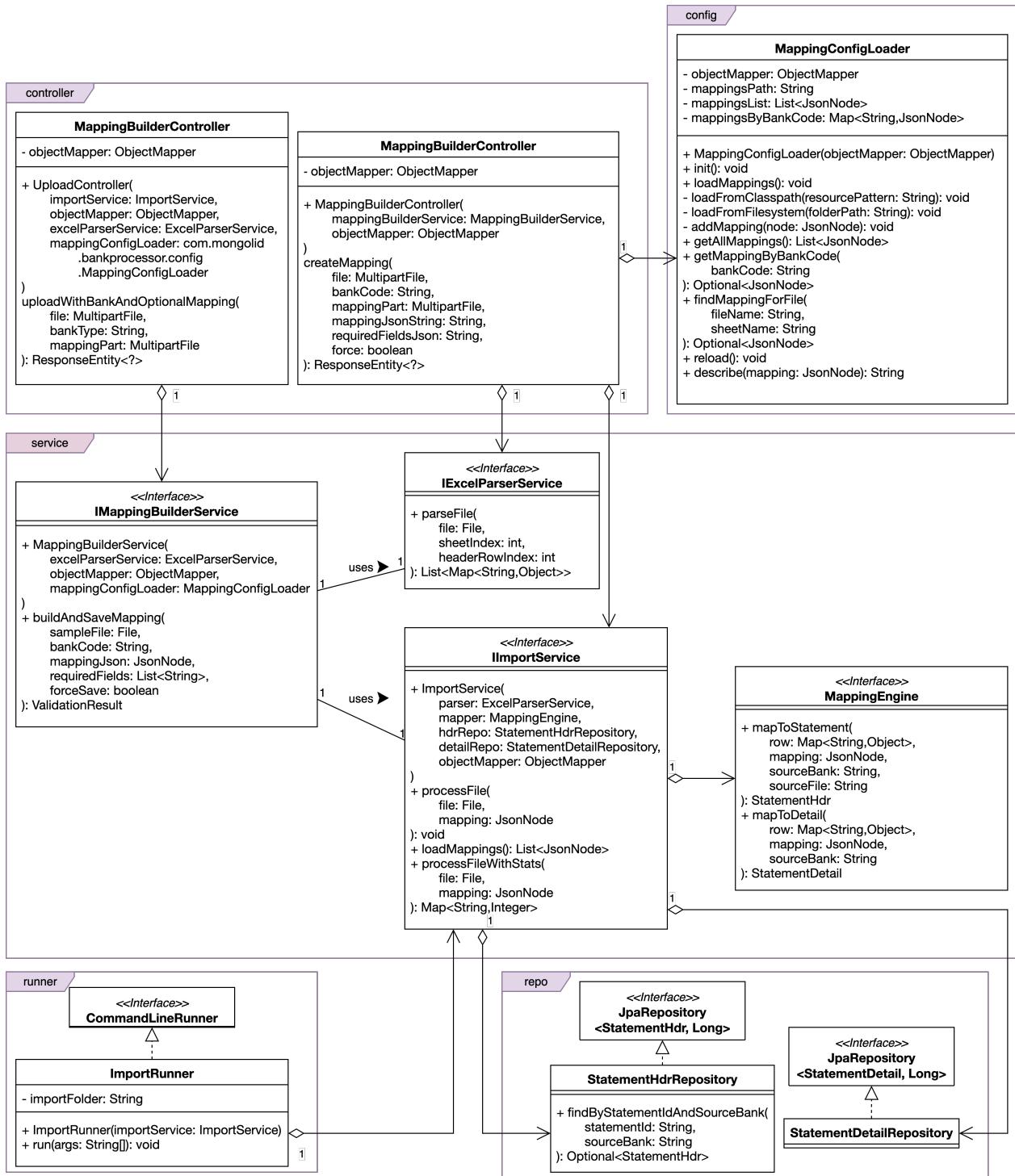
Зураг 6.3: "Builder" үлгэр загварын зохиомжийн диаграм



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

6.2. MINUPOS СИСТЕМИЙН БАНКНЫ EXCEL ХУУЛГЫГ БОЛОВСРУУЛАХ МОДУЛИЙН ЗОХИОМЖ

БҮЛЭГ 6. ДАДЛАГЫН ЯВЦ



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

6.3 MinuPOS системийн банкны Excel хуулгыг боловсруулах модулийн хэрэгжүүлэлт

7. ДҮГНЭЛТ

7.1 Yр дүн

7.1.1 Аймаг сүмдүн мэдээлэл авдаг форм

Хот/Аймаг: Булган ▾
Сум/Дүүрэг: Бугат ▾
Баг/Хороо: 03 ▾

Зураг 7.1: Формын эхний хувилбар

Хот/Аймаг:

Улаанбаатар

Сум/Дүүрэг:

Дүүрэг 2

Баг/Хороо:

Сонгох

Бүртгүүлэх

Зураг 7.2: Material ui болон React Select ашигласан байдал

7.1.2 *Toast компонент*

		Хямдрал							ХЯМДРАЛ ҮҮСГЭХ
Нүүр									
Дэлгүүр засах									
Бүтээгдэхүүн	▲	<input type="text" value="Нэр"/> 🔍							
Коллекции									
Бүтээгдэхүүн									
Хямдрал		Нэр	Эхлэх	Дуусах	Хямдрал	Төрөл	Лимит	Ашиглагдсан	Идэвхи
Бүтээгдэхүүний төрөл		Супер хямдрал 2	2021-08-02 10:00	2021-10-03 08:57	20	Хувь	1500	0	ҮГҮЙ
Банкны данс		Тархиа цэнэглээ аян 2	2021-08-04 09:01	2021-08-19 14:06	20	Хувь	100	0	ҮГҮЙ
Тохиргоо		Шинэ хямдрал	2021-08-05 11:19	2021-09-05 11:18	2000	Хямдрал	100	0	ТИЙМ
		Номын баар	2021-07-13 09:12	2021-09-03 01:51	2000	Хямдрал	90	0	ТИЙМ
		Хичээлийн шинэ жилийн мэнд хүргээ	2021-08-05 14:09	2021-08-31 09:45	123	Хямдрал	1000	0	ТИЙМ

Зураг 7.3: Хүсэлт амжилттай болсон үед харагдах Toast

Зураг 7.4: Хүсэлт амжилтгүй болсон үед харагдах Toast

7.2 Үр дүнгийн тайлан

Миний бие Хуур Мюзик Групп ХХК-д 21 хоногийн хугацаатай мэргэжлийн дадлагыг амжилттай гүйцэтгэж дуусгалаа. Уг хугацаанд хичээлийн хүрээнд үзсэн онолын ойлголтуудыг практик дээр туршиж, хэрэгжүүлсэн ба хөгжүүлэлт голчилсон технологийн компанийн ерөнхий үйл ажиллагаа, баг хооронд зохицон ажиллах чадвар, хөгжүүлэлтийн шинэ арга барилуудыг амжилттай эзэмшсэн гэж дүгнэж байна.

Continuous Integration/Continuous Deployment, GIT дээрх Feature Branch, ашиглаж буй програмчлалын хэлнийхээ давуу талыг судлан уг хэлээрээ сэтгэж бичих, том асуудлыг олон болгон хувааж багаар, алхам дэс дараатай асуудлыг шийдвэрлэх мөн ашиглаж буй сан, технологийнхоо гарын авлага буюу documentation-тай илүү сайн танилцаж уг технологийнхоо цаана нь буй концепцийг хялбараар ойлгох гэх мэт чадваруудыг эзэмшсэн. Үүнийгээ цаашид илүү хөгжүүлж мэргэшсэн фронт-энд хөгжүүлэгч болохоор зорьж байна. Дадлага хийсэн компани маань хэрэгжүүлж буй төсөлдөө үргэлж технологийн шинэ туршилтын хувилбаруудыг төвөгшөөлгүйгээр хэрэглэж, түүнийхээ алдааг илрүүлж, ажлын бус цагаараа хамтдаа шийдлийг хайж олон улсын нээлттэй эхийн төсөлд гар бие оролцдог нь бусад компаниудаас онцлог. Үүний үр дүнд манай дадлагын удирдагч болох С. Дөлмандах нь React-Native-н core contributor болж, 2019 онд болсон React-Native EU гэх олон улсын хөгжүүлэгчдийн эвентэд илтгэл тавьж байсан удаатай. Би цаашид өөрийн чөлөөт цагаа ашиглан дотоодын болон олон улсын нээлттэй эхийн төсөлд хувь нэмрээ оруулж гадны чадварлаг хөгжүүлэгчдийн арга барил, код бичих туршлага, тухайн асуудлыг хэрхэн шийдсэн гэх мэт үнэтэй мэдлэгүүдийг хуримтлуулж бусад хүмүүст мөн нээлттэй эхийн төсөлд оролцохын давуу талуудыг танилцуулж уриалахаар төлөвлөсөн байгаа.

Дадлагын эхэн үед React болон Next.js технологийн талаар судлах, түүнийгээ хэрэгжүүлэх, хөгжүүлэлтийн арга барилуудтай танилцах зорилготой байсан ба цаашид мэдээлэл технологийн ямар чиглэлээр мэргэшиж, түүндээ хүрэхийн тулд хэрхэн чадварлаг болох ёстойг ойлгосон тул зорилгодоо бүрэн хүрсэн гэдэгт итгэлтэй байна.

Bibliography

- [1] Declarative програмчлал болон Imperative програмчлалын ялгаа
<https://codeburst.io/declarative-vs-imperative-programming-a8a7c93d9ad2>
- [2] Material-ui Beta v5 хувилбарын Card ашиглах заавар
<https://next.material-ui.com/api/card/>

A. 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
40	menge	Quantity

Үргэлжнэ

Table A.1 – Үргэлжлэл

№	Герман	Англи
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

B. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

B.1 Форм

B.1.1 *useReducer* болон *React-Select* сан ашигласан байдал

```
1 import { useState, useReducer } from "react";
2 import Head from "next/head";
3 import styles from "../styles/Home.module.css";
4 import Select from "react-select";
5 import address from "../data/address";
6
7 const SET_CITY = "city";
8 const SET_DISTRICT = "district";
9 const SET_WARD = "ward";
10
11 const reducer = (state, action) => {
12   switch (action.type) {
13     case SET_CITY:
14       return {
15       city: action.index,
16       district: null,
17       ward: null,
18     };
19     case SET_DISTRICT:
20       return {
21       ...state,
22       district: action.index,
23       ward: null,
24     };
25     case SET_WARD:
26       return {
27       ...state,
28       ward: action.index,
29     };
30     default:
31       return state;
32   }
33 };
34
35 export default function Home() {
36   const [state, dispatch] = useReducer(reducer, {
37     city: null,
38     district: null,
39     ward: null,
40   });
41
42   const handleChange = (index, type) => {
43     dispatch({ type: type, index });
44   };
45 }
```

```

45
46  const register = (e) => {
47    e.preventDefault();
48    console.log(state);
49  };
50
51  return (
52    <div className={styles.container}>
53      <Head>
54        <title>Create Next App</title>
55      </Head>
56
57      <main className={styles.main}>
58        <div>
59          <form className={styles.grid} onSubmit={register}>
60            <label>
61              <p> /:</p>
62              <Select
63                value={address.map((i, index) => ({ ...i, index }))[state.city]}
64                onChange={(e) => handleChange(e.index, SET_CITY)}
65                options={address.map((i, index) => ({ ...i, index }))}getOptionLabel={(option) => option.name}
66                getOptionValue={(option) => option.index}
67                placeholder=" "
68              />
69            </label>
70
71            <label>
72              <p> /:</p>
73              <Select
74                value={
75                  state.district != null
76                  ? address[state.city].districts.map((i, index) =>
77                      ({...i,
78                        index,
79                        })))[state.district]
80                  : []
81                }
82            <optionLabel={(option) => option.name}
83            <optionValue={(option) => option.index}
84            <optionLabel={(option) => option.name}
85            <optionValue={(option) => option.index}
86            <optionLabel={(option) => option.name}
87            <optionValue={(option) => option.index}
88            <optionLabel={(option) => option.name}
89            <optionValue={(option) => option.index}
90            <optionLabel={(option) => option.name}
91            <optionValue={(option) => option.index}
92            <optionLabel={(option) => option.name}
93            <optionValue={(option) => option.index}

```

```

94         placeholder=""      "
95     />
96   </label>
97
98   <label>
99     <p> /:</p>
100    <Select
101      value={
102        state.ward != null
103        ? address[state.city].districts[state.district].
104          wards.map(
105            (i, index) => ({ ...i, index })
106          )[state.ward]
107        : []
108      }
109      onChange={(e) => handleChange(e.index, SET_WARD)}
110      options={
111        state.district != null
112        ? address[state.city].districts[state.district].
113          wards.map(
114            (i, index) => ({ ...i, index })
115          )
116        : []
117      }
118      getOptionLabel={(option) => option.name}
119      getOptionValue={(option) => option.index}
120      placeholder=""      "
121    />
122   </label>
123   <button className={styles.registerBtn}      }></button>
124 </form>
125 </div>
126 </main>
127 );

```

B.1.2 Functional component дээр state ашигласан байдал

```

1 import { useEffect, useState } from "react";
2 import Head from "next/head";
3 import styles from "../styles/Home.module.css";
4 import axios from "axios";
5 import useForm from "../utils/useForm";
6
7 const fetchData = (url) => {
8   //get data from next.js api
9   return axios
10    .get(`http://localhost:3000/api/${url}`)
11    .then((res) => {
12      const results = res.data;

```

```

13     return results;
14   })
15   .catch((err) => {
16     console.error(err);
17   });
18 };
19
20 export default function Home() {
21   const [values, handleChange] = useForm();
22   const [data, setData] = useState({
23     cities: [],
24     districts: [],
25     wards: [],
26   });
27
28   useEffect(() => {
29     fetchData(`cities`)
30       .then((res) => {
31         setData({ ...data, cities: res });
32       })
33       .catch((err) => {
34         console.error(err);
35       });
36   }, []);
37
38   const register = (e) => {
39     e.preventDefault();
40     console.log(values);
41   };
42
43   const handleSelect = (id, type) => {
44     if (type == "city") {
45       fetchData(`cities/${id}`)
46         .then((res) => {
47           setData({ ...data, districts: res, wards: [] }); //set
48             districts and clear wards data
49         })
50         .catch((err) => {
51           console.error(err);
52         });
53     } else if (type == "district") {
54       //get wards
55       fetchData(`cities/${values.city}/${id}`)
56         .then((res) => {
57           setData({ ...data, wards: res });
58         })
59         .catch((err) => {
60           console.error(err);
61         });
62     }
63   };

```

```

64  const OptionItems = (props) => {
65    const options = props.items.map((item) => {
66      return (
67        <option key={item.id} value={item.id}>
68          {item.name}
69        </option>
70      );
71    });
72
73    return options;
74  };
75
76  return (
77    <div className={styles.container}>
78      <Head>
79        <title>Create Next App</title>
80      </Head>
81
82      <main className={styles.main}>
83        <div>
84          <form className={styles.grid} onSubmit={register}>
85            <label>
86              /:
87              <select
88                value={values.city}
89                defaultValue="default"
90                name="city"
91                onChange={(e) => {
92                  handleChange(e.target.name, e.target.value);
93                  handleSelect(e.target.value, "city");
94                }}
95              >
96                <option value="default" hidden>
97                  /
98                </option>
99
100               {data.cities.length > 0 ? (
101                 <OptionItems items={data.cities} />
102               ) : null}
103
104             {/* {data.cities.map((item) => {
105               return (
106                 <option key={item.id} value={item.name}>
107                   {item.name}
108                 </option>
109               );
110             })} */}
111           </select>
112         </label>
113
114         <label>
115           /:

```

```

116     <select
117       value={values.district}
118       defaultValue="default"
119       name="district"
120       onChange={(e) => {
121         handleChange(e.target.name, e.target.value);
122         handleSelect(e.target.value, "district");
123       }}
124     >
125       <option value="default" hidden>
126         /
127       </option>
128       {data.districts.length > 0 ? (
129         <OptionItems items={data.districts} />
130       ) : null}
131     </select>
132   </label>
133
134   <label>
135     /:
136     <select
137       value={values.ward}
138       defaultValue="default"
139       name="ward"
140       onChange={(e) => {
141         handleChange(e.target.name, e.target.value);
142       }}
143     >
144       <option value="default" hidden>
145         /
146       </option>
147       {data.wards.length > 0 ? (
148         <OptionItems items={data.wards} />
149       ) : null}
150     </select>
151   </label>
152   <button style={{ margin: "5px" }}></button>
153 </form>
154 </div>
155 </main>
156 </div>
157 );
158 }

```

B.2 Toast компонент

B.2.1 Toast/context.tsx - Context үүсчэх

```

1 import { createContext, useReducer } from "react";
2

```

```

3  import { ActionType } from "types";
4
5  export interface ToastType {
6    id?: string;
7    message: string;
8    type: "success" | "info" | "warning" | "error";
9    duration?: number;
10 }
11
12 export const ACTION_TOAST = "TOAST";
13 export const ACTION_RESET = "RESET";
14 export const ACTION_CLEAR = "CLEAR";
15
16 export const INITIAL_STATE: { toasts: ToastType[] } = {
17   toasts: [],
18 };
19
20 type Actions =
21   | (ActionType<typeof ACTION_TOAST> & { toast: ToastType })
22   | ActionType<typeof ACTION_RESET>
23   | (ActionType<typeof ACTION_CLEAR> & { id: string });
24
25 export const ToastContext = createContext(null);
26 ToastContext.displayName = "ToastContext";
27 export const ToastControlContext = createContext(null);
28 ToastControlContext.displayName = "ToastControlContext";
29
30 function toast(state: typeof INITIAL_STATE, toast: ToastType): typeof
31   state {
32     return {
33       ...state,
34       toasts: [...state.toasts, toast],
35     };
36   }
37
38 function clear(state: typeof INITIAL_STATE, toastId: string): typeof
39   state {
40     return {
41       ...state,
42       toasts: state.toasts.filter((toast) => toast.id !== toastId),
43     };
44 }
45
46 function reducer(state = INITIAL_STATE, action: Actions): typeof state
47 {
48   switch (action.type) {
49     case ACTION_TOAST:
50       return toast(state, action.toast);
51     case ACTION_RESET:
52       return INITIAL_STATE;
53     case ACTION_CLEAR:
54       return clear(state, action.id);

```

```

52     default:
53       return state;
54   }
55 }
56
57 export const ToastProvider = ({ children }) => {
58   const [state, dispatch] = useReducer(reducer, INITIAL_STATE);
59
60   return (
61     <ToastContext.Provider value={state}>
62       <ToastControlContext.Provider value={dispatch}>
63         {children}
64       </ToastControlContext.Provider>
65     </ToastContext.Provider>
66   );
67 };

```

B.2.2 *Toast/hooks.ts - Custom hook бичих*

```

1 import { useCallback, useContext, useMemo } from "react";
2
3 import {
4   ACTION_CLEAR,
5   ACTION_TOAST,
6   ACTION_RESET,
7   ToastContext,
8   ToastControlContext,
9   ToastType,
10 } from "./context";
11
12 function generateToastId() {
13   return Math.random().toString(36).substr(2, 9);
14 }
15
16 export function useToast() {
17   const state = useContext(ToastContext);
18   if (!state) throw new TypeError("Please use within ToastProvider");
19   return state;
20 }
21
22 export function useToastControl() {
23   const dispatch = useContext(ToastControlContext);
24   if (!dispatch) throw new TypeError("Please use within ToastProvider");
25   ;
26
27   const reset = useCallback(() => dispatch({ type: ACTION_RESET }), [
28     dispatch]);
29
30   const toast = useCallback(
31     (toast: ToastType) => {
32       const toastId = generateToastId();
33
34       dispatch({
35         type: ACTION_TOAST,
36         id: toastId,
37         toast,
38       });
39
40       return toastId;
41     },
42     [dispatch]
43   );
44
45   return { toast, reset };
46 }
47
48 export function useClear() {
49   const dispatch = useContext(ToastControlContext);
50
51   const clear = useCallback(() => dispatch({ type: ACTION_CLEAR }), [
52     dispatch]);
53
54   return clear;
55 }
56
57 export function useReset() {
58   const dispatch = useContext(ToastControlContext);
59
60   const reset = useCallback(() => dispatch({ type: ACTION_RESET }), [
61     dispatch]);
62
63   return reset;
64 }
65
66 export function useControl() {
67   const dispatch = useContext(ToastControlContext);
68
69   const control = useCallback((action) => dispatch(action), [
70     dispatch]);
71
72   return control;
73 }
74
75 export function useToastId() {
76   const state = useContext(ToastContext);
77
78   const toastId = useState(() => generateToastId());
79
80   const toast = useCallback(
81     (toast: ToastType) => {
82       const toastId = toastId[1];
83
84       dispatch({
85         type: ACTION_TOAST,
86         id: toastId,
87         toast,
88       });
89
90       return toastId;
91     },
92     [dispatch]
93   );
94
95   return { toast, toastId };
96 }
97
98 export function useControlId() {
99   const dispatch = useContext(ToastControlContext);
100
101   const controlId = useState(() => generateToastId());
102
103   const control = useCallback((action) => dispatch(action), [
104     dispatch]);
105
106   return { control, controlId };
107 }

```

```

31     dispatch({ toast: { ...toast, id: toastId }, type: ACTION_TOAST
32             });
33
34     setTimeout(() => {
35         dispatch({ id: toastId, type: ACTION_CLEAR });
36     }, toast.duration || 4000);
37 },
38 [dispatch]
39 );
40
41 return useMemo(() => {
42     return { reset, toast };
43 }, [toast, reset]);
44 }

```

B.2.3 Toast/index.ts

```

1 export * from "./Toast";
2 export * from "./context";
3 export * from "./hooks";

```

B.2.4 Toast/Toast.tsx - Үндсэн Toast компонент

```

1 import { useState, forwardRef } from "react";
2
3 import { IconButton, Snackbar } from "@material-ui/core";
4 import MuiAlert, { AlertProps } from "@material-ui/core/Alert";
5 import { Close as CloseIcon } from "@material-ui/icons";
6
7 import { ToastType } from "./context";
8 import { useToast } from "./hooks";
9
10 const Alert = forwardRef<HTMLDivElement, AlertProps>(function Alert(
11     props,
12     ref
13 ) {
14     return <MuiAlert elevation={6} ref={ref} variant="filled" {...props}>
15         />;
16 });
17
18 export function Toast({ message, type, duration = 4000 }: ToastType) {
19     const [open, setOpen] = useState(true);
20
21     const onClose = () => setOpen(false);
22
23     const action = (
24         <IconButton
25             size="small"
26             ...
27         >
28             ...
29         </IconButton>
30     );
31
32     return (
33         <Snackbar open={open} autoHideDuration={duration} onClose={onClose}>
34             <Alert type={type}>{message}</Alert>
35         </Snackbar>
36     );
37 }

```

```
25     aria-label="close"
26     color="inherit"
27     onClick={onClose}
28   >
29     <CloseIcon fontSize="small" />
30   </IconButton>
31 );
32
33 return (
34   <Snackbar
35     anchorOrigin={{ horizontal: "center", vertical: "bottom" }}
36     open={open}
37     autoHideDuration={duration}
38     onClose={onClose}
39     action={action}
40   >
41     <Alert onClose={onClose} severity={type} sx={{ width: "100%" }}>
42       {message}
43     </Alert>
44   </Snackbar>
45 );
46 }
47
48 export function ToastContainer() {
49   const { toasts } = useToast();
50   return toasts.map((toast) => <Toast key={toast.id} {...toast} />);
51 }
```

B.3 Toast компонент дээр интерфэйсийн автоматжуулсан тест бичсэн байдал

B.3.1 tests/Toast.test.tsx

```
1 import {
2   cleanup,
3   render,
4   screen,
5   waitForElementToBeRemoved,
6 } from "@testing-library/react";
7 import userEvent from "@testing-library/user-event";
8
9 import { Toast } from "../../components/Toast";
10
11 jest.useFakeTimers();
12 describe("Test: Toast component", () => {
13   test("show message and delete toast after close button", async () =>
14     {
15       const message = "Hello, it's toast";
16
17       render(<Toast message={message} type="success" />);
18       expect(screen.getByText(message)).toBeInTheDocument();
19
20       userEvent.click(
21         screen.getByRole("button", {
22           name: /close/i,
23         })
24       );
25
26       await waitForElementToBeRemoved(() => screen.getByText(message));
27       expect(screen.queryByText(message)).not.toBeInTheDocument();
28     });
29
30   test("duration", async () => {
31     const cssAnimation = 300;
32     const toasts = [
33       {
34         duration: 4000,
35         message: "Toast_1",
36       },
37       {
38         duration: 5000,
39         message: "Toast_2",
40       },
41     ];
42
43     toasts.forEach((toast) =>
44       render(
45         <Toast
46           message={toast.message}
```

```

46         type="success"
47         duration={toast.duration}
48     />
49   )
50 );
51
52 expect(screen.getByText("Toast\u201d1")).toBeInTheDocument();
53 await waitForElementToBeRemoved(() => screen.getByText("Toast\u201d1"),
54   {
55     timeout: toasts[0].duration + cssAnimation,
56   });
57 expect(screen.queryByText("Toast\u201d1")).not.toBeInTheDocument();
58
59 expect(screen.getByText("Toast\u201d2")).toBeInTheDocument();
60 await waitForElementToBeRemoved(() => screen.getByText("Toast\u201d2"),
61   {
62     timeout: 1000 + cssAnimation,
63   });
64 expect(screen.queryByText("Toast\u201d2")).not.toBeInTheDocument();
65 });
66
67 test("check\u201dtype", () => {
68   const types = [
69     {
70       class: "filledSuccess",
71       message: "Success\u201dtoast",
72       type: "success",
73     },
74     { class: "filledError", message: "Error\u201dtoast", type: "error" },
75     { class: "filledInfo", message: "Info\u201dtoast", type: "info" },
76     {
77       class: "filledWarning",
78       message: "Warning\u201dtoast",
79       type: "warning",
80     },
81   ];
82
83   types.forEach((opt) => {
84     const { container } = render(
85       <Toast
86         message={opt.message}
87         type={opt.type as "success" | "info" | "warning" | "error"}
88       />
89     );
90     const alert = container.firstChild;
91
92     expect(alert.firstChild).toHaveClass(`MuiAlert-${opt.class}`);
93     cleanup();
94   });
95 });
96 });

```