МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

# Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему «Проектування інформаційної системи побудови 2D/3D лекал»

«Development of a 2D/3D pattern building information system»

Виконав: студент денної форми навчання
спеціальності 126 – Інформаційні системи та технології .

(шифр і назва напряму підготовки, спеціальності)

Жмакіна Анастасія Семенівна

(прізвище, ім'я, по-батькові)

Керівник д.т.н., проф. Малахов Є.В

(науковий ступінь, вчене звання, прізвище та ініціали, підпис)

Рецензент: к.т.н., доц. Пенко В.Г.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2022 р.

Завідувач кафедри
_____ _Євгеній МАЛАХОВ_
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «___» ____ 2022 р.
Оцінка _____/____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК
_____ _Володимир ВИЧУЖАНІН_
(підпис) (ім'я, прізвище)

Одеса – 2022

# АНОТАЦІЯ

Пошив одягу завжди був невід’ємною часткою культури суспільства. Однак, останні 7 років швейна майстерність набула популярності та пристрасті не тільки як професія, але як і хобі. Через те, що конструювання займає провідну роль при створенні одягу, яка є точною та потребує певних знань, є потреба у сторонній допомозі чи ресурсах.

Метою дипломною роботи є розробка десктопного застосунка для будування викрійок одягу.

У функціональність застосунка входить:

– отримання теоретичної інформації, пов’язаної із шиттям;

– конструювання базових викрійок одягу на основі індивідуальних показників фігури людини;

– 3D-модель готової викрійки;

– отримання готового креслення у форматі PDF для можливості друкування.

Для розробки використані Blender, мова програмування C# в рамках графічної підсистеми WPF.

# ABSTRACT

Tailoring has always been an essential part of the society's culture. However, in the last seven years, sewing has become popular and passion not only as a profession, but also as a hobby. Since construction maintains a leading role in tailoring that requires specific knowledge in this sphere, so that people have a need for outside help or resources.

The aim of the work is to develop a desktop app for making clothing pattern.

The functionality of the app includes:

– obtaining theoretical information related to sewing;

– designing basic clothing patterns based on individual measurements of the human figure;

– 3D - model created pattern;

– saving constructed pattern in PDF format for printing.

For the development used Blender, the programming language C# within graphical subsystem WPF.

**ЗМІСТ**

## ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

МДУДТ – Московський державний університет дизайну та технологій.

СЛ – сантиметрова стрічка.

ЦНДІШП – Центральний науково-дослідницький інститут швейної промисловості.

MVVM (Model-View-ViewModel) – шаблон проектування.

UWP – Universal Windows Platform.

WPF – Windows Presentation Foundation.

# ВСТУП

Ще з давнини людство почало вдосконалювати ремесло створювання різноманітного одягу. Спочатку люди замість нього використовували різноманітні природні матеріали: шкіри тварин, листя, кору дерев та інше. Далі було освоєно мистецтво прядіння, плетіння та ручне створювання тканин. Перші освідомленні спроби скроїти одяг за формою людської фігури, а не лише драпірувати матерію, виникли у XII ст. Західної Європі. Від тоді почався шлях розвитку швейної промисловості з простого плаття на шнурівці до використання комп'ютерних технологій, котрі допомагають виготовляти сучасну одежу.

Наприкінці XX століття почався спад зацікавленості виробу убрання власноруч серед звичайних жителів. Однак, останні сім років за даними Google Trends [1] спостерігається зріст популярності та інтересу до швейної майстерності не тільки у миленіалів, а й у молоді. Для цього існує декілька факторів, котрі сприяли відновленню цього ремесла. По-перше, люди почали втомлюватися від одноманітного масмаркетового одягу, серед котрого доволі тяжко знайти потрібний розмір під нестандартні типи фігури. По-друге, з росту популярності супергероїв та комп'ютерних ігор з'явилась така течія як косплей, що дозволила залучитись інтересом підлітків. По-третє, збільшення популярності екофрендлі руху та критики швидкої моди, так як в розвинутих країнах виробляють величезну кількість відходів, купуючи та викидаючи одяг, при виробництві якого кожний етап несе шкоду водної, наземної та атмосферної екосистем. Також ще одним впливаючим фактором став COVID-19, коли багато людей почали дивитися навчальні ресурси та намагатися виробляти власноруч убрання [2, 3].

У зв'язку з тим, що цінність навичок шиття в даний час привертає до себе підвищену увагу, на мою думку, можна припустити, що відродження ремеслу буде набирати оберти та більш людей будуть починати або продовжувати виготовляти власний одяг, маски для обличчя, різноманітні

костюми для косплею та багато іншого. Також із великої кількості переваг, шиття особливо добре підходить для цих незвичайних часів, тому що цей процес має заспокійливий ефект та знімає стрес, розвиває креативність, допомагає зануритися у дивний вимір мистецтва та відкрити у собі навички дизайнеру. Ще можна згадати, що шиття допомагає розвинути дрібну моторику, покращує увагу та концентрацію, а також вчить важливості терпіння та самоконтролю [4].

Але через те, що конструювання займає провідну роль при виготовленні виробу, та не кожний готовий приділяти багато часу цьому або не має певних знань в даної галузі, є потреба у сторонній допомозі чи ресурсах будування базових лекал. Також сучасні методики конструювання мають велику кількість розрахункових формул вимірів фігури та навчальні ресурси мають сухий професійний стиль оповідання, що ускладнює освоєння тексту та надалі приводить до помилок, котрі впливають на коректність процесу побудови викрійок.

На даний момент більшість програм для конструювання різноманітних викрійок одежі мають великий функціонал та свої переваги. Серед десктопних продуктів, які надають можливість конструювати одяг, варто виділити САПР GRAFIS, САПР Julivi, Assyst, Cameo. Однак, дані програми більш орієнтуються на велике швацьке виробництво ніж для індивідуального використання: люди не можуть собі дозволити придбати дорогий інструментарій або він дуже складний та не зрозумілий. В наслідок цього ми спостерігаємо розвиток наступних «тривог» користувачів:

1) відсутність необхідних ресурсів для конструювання одягу;

2) висока ціна аналогів та велика складність освоєння для особистого користування;

3) сумніви у своїх здібностях та навичках конструювання одягу.

На жаль, наведені раніше продукти не дають можливості вирішити усі проблеми, які описувалися вище.

Проаналізувавши предметну область, можна сформулювати наступний необхідний функціонал для таких систем:

1) отримання необхідної теоретичної інформації щодо методів конструювання одягу, зняття індивідуальних розмірних ознак та алгоритму конструювання деталей одягу;

2) будування креслення базових конструкцій на індивідуальну фігуру, нарощування припусків та оформлення зрізів;

3) отримання готової викрійки у форматі PDF;

4) 3D-моделювання виробу.

Виходячи з цього, сформульована мета роботи – розробка системи для будування базових лекал. Одним з таких базових лекал є лекало спідниці за методикою Центрального науково-дослідного інституту швейної промисловості (ЦНДІШП), яка для цього проекту була обрана першорядною для реалізації.

# 1 ПОСТАНОВКА ЗАДАЧІ

Мета роботи – розробити систему для конструювання викрійок у 2D та 3D. Одним з таких базових лекал є лекало спідниці, що й реалізовано в данному проекту.

Для досягнення ключових результатів потрібно вирішити наступні підзадачі:

1) порівняти методики та програмні продукти для конструювання одягу;

2) проаналізувати та обґрунтувати вибір технологій, потрібних для реалізації системи;

3) розробити алгоритм побудови викрійки юбки;

4) спроектувати та реалізувати додаток для побудови викрійок.

Ще одним важливим пунктом у досягненні мети являється здобуття теоретичних та практичних навичок побудови основних лекал одягу згідно з методикою ЦНДІШП, які можна отримати на курсах крою та шиття під керівництвом досвідчених швейних майстрів.

# 2 ОГЛЯД АНАЛОГІВ

## 2.1 Порівняння програмних продуктів, що реалізують конструювання викрійок одягу

Конструювання одягу є досить важким до освоєння процес, що потребує уваги. І для можливості конструювання одягу новачкам, а також для збільшення продуктивності працівників індивідуального пошиття, розроблено системи автоматизованого проектування. САПР – це загальна класифікація всім конструкторських програм. Основні функції САПР схожі, але різняться зовнішній вигляд, додатковий функціонал та ціна.

На даний момент більш популярними десктопними продуктами, які надають можливість конструювати одяг, є САПР GRAFIS, САПР Julivi, Assyst, Cameo.

Далі буде розглянуто основні переваги та недоліки програм.

### 2.1.1 САПР GRAFIS

САПР Графіс – це один із найперших і досить часто використовуваних інструментів для створення моделей у Росії, випущена ще 1991 року російської компанією Кадрус [5].

У програму внесені різні методики конструювання (Optimass, Мюллер та син, ЦОТШЛ, ЄМКО СЕВ). САПР включає варіанти основних виробів (спідниці, штани, чоловічі і жіночі плечові основи та інше). Також, перевагою є вміння виконувати автоматичну градацію за розмірними ознаками, задавати припуски виробів та робити ручне або автоматичне розкладання деталей крою.

Повна вартість пакету «Конструювання та моделювання одягу» – 120 000 грн., а оренда на рік – 24 000 грн. Однак, даний продукт не надає можливість ознайомиться з обмеженим інструментарієм у вигляді демоверсії.

Ще один з недоліків – це відсутність можливості конвертувати готове креслення у формат PDF для подальшої печаті. Однак, є можливість обмінюватися інформацією з різними партнерами за допомогою форматів даних AAMA-DXF та HPGL.

### 2.1.2 САПР Julivi

Компанія САПРЛЕГПРОМ розробник програмного забезпечення JULIVI. Займає одну з провідних позицій на ринку інформаційних технологій з 1980-х років [6].

Комплекс програм «Конструктор одягу» призначений для конструювання лекал одягу, головних уборів на швейному виробництві.

САПР підтримує розробку 2D и 3D проектування одягу за різними методиками конструювання в одному або декілька розмірів.

Має такі можливості:

- побудова креслення конструкції;
- нарощування припусків та оформлення зрізів;
- автоматична градація;
- виведення лекал на друк.

Вартість комплексу складає «Конструктор одягу» 1750 Євро.

На офіційному сайті компанії САПРЛЕГПРОМ є демонстраційна версія, де, підключившись на їх сервер, можна випробувати деякий функціонал.

### 2.1.3 Assyst

Німецька програма Assyst належить компанії Assyst GmbH. На його базі працює близько 40% німецьких швейних підприємств та понад 200 світових брендів.

З 2009 року компанія є офіційним представником CAD Assyst в Україні та Білорусі, Росії, Естонії, Литві [7].

САПР складається з низки окремих модулів: конструювання, 3D-моделювання, розкладки лекал, друк, конвертація, оптимізація розкрою, управління даними, виробництвом та інших. Пакет модулів складається залежно від вимог виробництва та від цього залежить ціна.

Програма для конструювання одягу дозволяє вирішувати такі виробничі завдання:

– розробка з нуля та моделювання одягу нових фасонів;

– створення необхідних за розміром лекал в автоматичному режимі;

– оптимізація виробничих замовлень та формування інструкцій для розкрійного цеху.

Програма імпортує та експортує файли у форматі DXF.

### 2.1.4. Cameo

Американська компанія Wild Ginger Software спеціалізується на розробці програмного забезпечення та супутніх товарів для дизайнерів одягу, любителів шиття та рукоділля, а також для викладачів та студентів з галузі дизайну моди та театрального костюма [8].

Програма для професійного дизайну Cameo – це потужне програмне забезпечення для створення візерунків, класифікації, виготовлення за розмірами та створення маркерів для дизайнерів одягу, які включають у свої можливості побудови, моделювання та градації лекал та експорт в стандартний формат AutoCAD DXF.

Вартість продукту складає приблизно 950 доларів за кожний блок, а для ознайомлення з основними функціями є демоверсія.

Також, на сайті можна придбати книги англійською по конструюванню та моделюванню одягу.

### 2.1.5 Висновки

Проаналізувавши розглянуті вище програми зроблені наступні висновки:

1) з чотирьох описаних систем, тільки Cameo має прийнятну ціну та мінімальний функціонал, необхідний для індивідуального використання;

2) більшість продуктів орієнтовано на велике швейне виробництво, що обумовлюється нагромадженим функціоналом та дорожнечею;

3) тільки два з чотирьох програмних забезпечень, котрі мають демоверсію за якою можна оцінити та випробувати обмежений функціонал.

## 2.2 Порівняння методик конструювання одягу

Естетичне сприйняття готового одягу залежить від точної побудови креслення основи виробу та вміння правильно використати його при конструюванні. Поважне та копітке вивчення методик побудови основ виробу є базою в освоєнні грамотної побудови креслення на конкретну фігуру.

Існує 2 класи методик, які поділяються за характером вихідної інформації:

1) методи першого класу (розрахунковий метод конструювання) основані на використанні дискретних значень типових фігур, даних о збільшеннях та спосіб його формоутворення. Методи ЦНДІШП, «Мюллер та син», МГУДТ та інші відносяться до цього класу;

2) методи другого класу основані на прямих вимірах оболонки і поверхні зразка, що розгортається. До них відносяться методи тріангуляції, допоміжних ліній розгортання, січних площин та геодезичних ліній.

Кожен з розрахункових методів представляє собою послідовність розрахункових формул, в які входять величини вимірювання фігури та збільшення.

Для розрахунку креслярської основи конструкцій в проекті обрана єдина методика конструювання Центрального науково-дослідницького інституту швейної промисловості. Вона передбачає розгортку фігури на площині, заснована на вимірах людини, надбавках на вільне облягання та математично обґрунтованих формулах. Система отримала назву - Єдина методика конструювання. Також методика багаторазово перевірена та відповідає наступним вимогам [9,10]:

1) надає можливість побудувати с однакої точністю креслення деталей виробу різноманітних розмірів, росту та повнот;

2) позволяє получити гарну посадку виробу;

3) ефективна та точна методика;

4) можливість отримання модних ліній та крою без принципової перебудови креслення конструкції;

5) придає гарне формоутворення;

6) використовує розмірні ознаки жіночих типових фігур, закріплених сучасними розмірними стандартами.

# 3 ОБГРУНТУВАННЯ ВИБОРУ АРХІТЕКТУРИ, СТЕКУ ТЕХНОЛОГІЙ ТА ШАБЛОНУ ПРОЕКТУВАННЯ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ 2D/3D ПОБУДОВИ ЛЕКАЛ

## 3.1 Вибір архітектури

Сучасні системи зазвичай розділяють на 3 логічних шари:

1) шар представлення – відповідає за відображення даних користувачеві, а також надає інструменти для зручного маніпулювання даними;

2) шар бізнес-логіки – містить правила, залежності предметної області та деякі програмні реалізація для вирішення поставлених задач;

3) шар моделі – надає дані, що будуть оброблятися на етапі вирішення задач.

Для реалізації десктопного застосунку обрана багаторівнева архітектура. Ця архітектура використовується для структурування програм, які можна розкласти на групи підзадач, що є на певних рівнях абстракції. Кожен шар надає послуги для наступного, вищого шару. Також вона пом'якшує зростаючу складність застосунків, спрощує та робить більш гнучким їх доопрацювання. Використовується для общин десктопних та веб-застосунків.

Серед переваг даної архітектури можна перерахувати:

1) шари багаторівневої архітектури спрощують стандартизацію, так як рівні чітко визначаються;

2) якщо зміни вносяться всередині одного шару архітектури, інші шари залишаються незмінними;

3) одним нижчим шаром архітектури можуть користуватися різні шари вищого рангу.

Серед мінусів – це не універсальність та пропуск деяких шарів при проектуванні.

### 3.2 Вибір шаблонну проектування

Серед шаблонів проектування використано шаблон MVVM (Model-View-ViewModel), який призначений для відділення коду інтерфейсу користувача від решти коду та розділяє систему на три рівня:

1) Model – відповідає за логіку роботи з даними та опис необхідних даних для роботи з застосунком;

2) View – відображає візуальні елементи та елементи керування на екрані. Зазвичай, являються підкласами UIView;

3) ViewModel або модель уявлення пов'язує модель та уявлення через механізм прив'язки даних. ViewModel також містить логіку з отримання даних з моделі, які потім передаються у View.

MVVM з’явився для обходу обмежень патернів MVC та MVP та маючий деякі з їх сильних сторін. Ця модель вперше з’явилась у составі фреймворка Small Talk в 80-х роках та пізніше був оновлений згідно з останніми покращеннями шаблону MVP [11].

За допомогою даного шаблону використовується прив’язка даних та встановлюється слабкий зв’язок, який забезпечує синхронізацію інтерфейсу користувача та зв’язаних даних, спрощення тестування і зміну різних блоків коду без шкоди для інших.

### 3.3 Вибір операційної системи для розробки

Система розроблюється під операційну систему Windows.

Операційна система Microsoft Windows є найпопулярнішою операційною системою завдяки простому інтерфейсу користувача та навігації (рис 3.1) [12]. ОС Windows краща за інші ОС по функціональності, єдності ОС, системним утилітам, базі підтримки та продуктивності.

Також, компанія Microsoft надає великий інструментарій для розробки систем під ОС Windows.

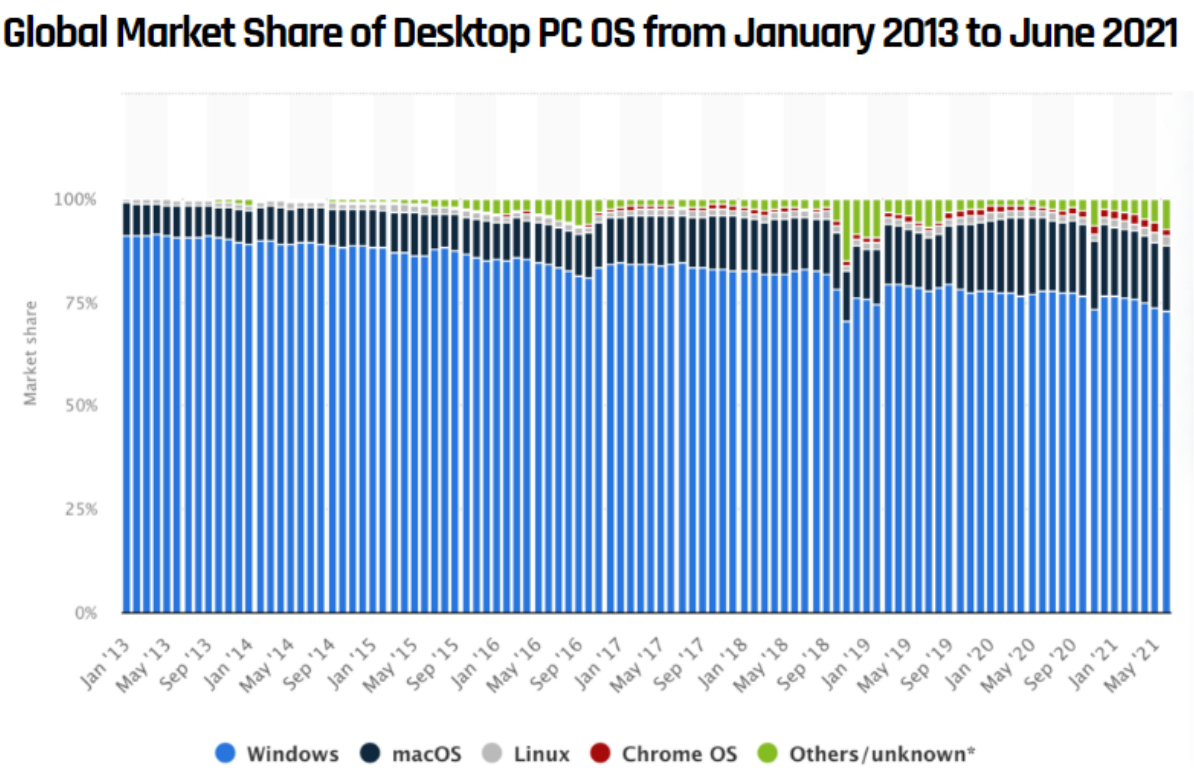


Рисунок 3.1 – Статистика використовування операційних систем з 2013 по 2021 рік

### 3.4 Вибір мови програмування

C# являється об'єктно-орієнтованою мовою з автоматичним керуванням пам'яттю, розроблена компанією Microsoft для платформи .NET [13], здатної запропонувати широкий спектр функціоналу, який доповнюється синтаксичними конструкціями. На C# створюють різноманітні програми для екосистеми Microsoft: веб-додатки, Standalone додатки, ігри з використуванням Unity, проекти для Microsoft Xbox.

Серед переваг мови можна перечислити: наявність ООП, кросплатформність, автоматична складання сміття та управління пам’яттю, велика кількість різноманітних бібліотек, що робить його більш швидкою та ефективною мовою програмування.

### 3.5 Вибір технологій для реалізації застосунку

Для розробки застосунків на .NET, що являється гарним вибором для написання програм під Windows, компанія Microsoft надає 3 очевидних варіанта вибору фреймворків [14]:

1) Windows Forms – це стабільна платформа для настільних додатків Windows, яка була представлена одночасно з .NET у 2002 році. WinForms дозволяє розробникам створювати досить складні програми, але має проблеми з гнучкістю та підтримкою високої роздільної здатності екрану;

2) Windows Presentation Foundation – друге покоління графічних інструментів Microsoft .NET. Являється одним з найпопулярніших варіантів розробки під обрану операційну систему. Заснований на графічному API DirectX, тому добре підходить для розробки 2D/3D графіки та спецефектів. Базова мова уявлення заснована на XAML, мові XML;

3) Universal Windows Platform (UWP) – еволюція нового фреймворка, представлена у 2012 році для Windows 10. Метою даної платформи є допомога у створенні універсальних додатків, які підтримують Windows 10, без змін у коді.

WPF та UWP дозволяють створювати більш сучасний користувацький інтерфейс та мають багато переваг, далі їх буде детальніше розглянуто.

Обидві системи мають багато переваг, таких як:

– для UI використовується мова – XAML, що дозволяє створити сучасні інтерфейси на будь-який смак;

– прив’язка даних, стосовно шаблону MVVM, забезпечує розділ інтерфейсу від шару бізнес-логіки;

– засновані на векторному графічному UI на основі DirectX, в той час як за малювання графіки в WinForms відповідає User32 и GDI+.

Але для розробки системи конструювання викрійок одягу була обрана платформа WPF, так як:

1) дозволяється створювати програми під велику кількість ОС сімейства Windows;

2) масштабованість під різну роздільну здатність екранів, так як усі елементи вимірюються у незалежних від пристрою одиницях;

3) хоча об’єм програм на Windows Presentation Foundation та споживання пам’яті вище, але це компенсується широкими графічними можливостями та продуктивністю при малюванні графіки;

4) велика та розвинута спільнота.

Для розробки 3D-моделі використовується професійне вільне та відкрите програмне забезпечення Blender. Продукт відкриває неперевершену свободу робочого процесу для розкадрувальників та 2D та 3D художників, що включає засоби моделювання, скульптингу, анімації, симуляції, рендерингу та багато іншого.

Для відображення 3D в WPF обрана 3D-библиотека с відкритим вихідним кодом Helix Toolkit, яка розповсюджується під ліцензією MIT. Бібліотека заснована на .NET і орієнтована на платформу WPF. Мета цієї бібліотеки полягає в тому, щоб спростити роботу з 3D у графічному інструменту WPF, а також надати функції, які не включені в стандартну візуальну 3D-модель WPF [15].

Для того, щоб конвертувати викрійки у формат PDF, застосовувався NuGet пакет Spire.PDF. Spire.PDF для .NET – це професійний API PDF, який використовується для створення, написання, редагування, обробки та читання PDF-файлів без будь-яких зовнішніх залежностей у .NET [16]. Використовуючи цю бібліотеку .NET PDF, можна реалізувати широкі можливості для створення PDF-файлів з нуля або обробки існуючих PDF-документів повністю за допомогою C#/VB.NET без встановлення Adobe Acrobat. Крім того, дана бібліотека може бути застосовувана для простого перетворення тексту, зображень, SVG, HTML у PDF та перетворення PDF в Excel за допомогою C#/VB.NET у високій якості. Існує платна та безкоштовна версії. При використанні безкоштовної версії та

створенні/оновлені нових PDF-документів додається на кожну сторінку напис – «Evaluation Warning: The document was created with Spire.PDF for .NET.», але це практично ніяк не впливає на отриманий результат.

Бібліотека iTextSharp використовується для отримання розміру різних форматів паперу (A4, A5 і т.д.), пропонує один з найбільш добре документованих і універсальних механізмів PDF, написаних на Java та .NET, та дозволяє інтегрувати функції PDF не тільки у робочий процес, але й у програми, процеси або продукти.

Для спрощення реалізації патерну MVVM обрана бібліотека Prism, яка надає функції, які допомагають з реалізацією шаблону в додатках. Ці функції втілюють найпоширеніші методи реалізації шаблона MVVM, котрі допомагають спростити реалізацію, призначені для підтримки можливості тестування та гарно працюють с Expression Blend и Visual Studio [17].

Для визначення елементів інтерфейсу користувача, прив’язки даних, створювання різноманітних стилів, використовується мова розмітки інтерфейсу користувача XAML (eXtensible Application Markup Language).

# 4 ПРОЕКТУВАННЯ СИСТЕМИ 2D/3D ПОБУДОВИ ЛЕКАЛ

## 4.1 Алгоритм конструювання спідниці

Вихідними даними для конструювання викрійки спідниці [18] є мірки фігури відносного розміру, визначальні ширину та довжину спідниці та конструктивні збільшення на вільне облягання.

Для будування креслення основи юбки використовується наступні індивідуальні мірки: обхват стегон і талії, довжину від талії до бажаної довжини (додаток А).

Сітку креслення прямої спідниці утворює рядком горизонтальних та вертикальних ліній. До горизонтальних відносяться лінії талії, стегон та низу; до вертикальних – лінії направлення бокового членування та лінії розташування виточок.

По-перш, розраховується ширина базисної сітки ($Ш_{бс}$), яка розраховується за формулою

$$Ш_{бс} = С_{б} + П_{б} \quad (4.1)$$

де $С_{б}$ – напівобхват стегон, см;

$П_{б}$ – збільшення по стегну, см.

Алгоритм побудування базисної сітки спідниці:

Крок 1. Визначаємо точку W – це початкова точка, звичайно розташована від 5-х сантиметрів зверху та зліва.

Крок 2. Положення лінії стегон: від точки W проводиться лінія перпендикулярно вниз от 18 до 20 см, залежно від висоти посадки сідниць та визначаємо точку Н. Далі від точки Н проводиться перпендикуляр вправо, довжиною $Ш_{бс}$ отримуємо точку $Н_{1}$.

Крок 3. Положення лінії низу: від точки W опускаємо перпендикуляр вниз рівний $Д_{юб}$ отримуємо точку В. Потім від знайденої точки проводиться перпендикуляр вправо, довжиною $Ш_{бс}$ , визначаємо точку $В_1$.

Крок 4. Ширина базисної сітки: від точки W проводиться перпендикуляр вправо рівний $Д_{юб}$ отримуємо точку $W_1$.

Крок 5. Положення лінії боку: для розрахунку положення точки $W_2$ використовується формула (4.2), але якщо живіт яскраво виражений – формула (4.3).

$$W_2 = \frac{TT_1}{2} \tag{4.2}$$

$$W_2 = \frac{TT_1}{2} - 1 \tag{4.3}$$

де $TT_1$ – відстань між точками Т та $Т_1$.

Крок 6. Сумарний розрахунок виточок визначається за виразом (4.4)

$$\sum В = Ш_{бс} - (С_т + П_т) \tag{4.4}$$

де $С_т$ – напівобхват талії, см;

$П_т$ – збільшення по талії, см.

Далі знаходиться виточки по спинці (Р.в.сп.) за (4.5), по боку (Р.в.б.) за (4.6) та поличці (Р.в.пер.) за (4.7).

$$Р.в.сп. = 30\% \sum В \tag{4.5}$$

$$Р.в.б. = 50\% \sum В \tag{4.6}$$

$$Р.в.пер. = 20\% \sum В \tag{4.7}$$

Крок 7. Побудова талієвої виточки по спинці: від точки $W_2$ відкладається вліво та вправо точки $W_{21}$ і $W_{22}$ на відстань $\frac{Р.в.сп.}{2}$.

$$W_3 = \frac{W\,W_{21}}{2} \tag{4.8}$$

Від точки $W_3$ (див. формулу (4.8)) відкладається вправо та вліво на відстань $\frac{Р.в.сп.}{2}$ та отримуємо точки $W_{31}$ і $W_{32}$ .

Від точки $W_3$ вниз опускається перпендикуляр рівний від 13 до 15 см отримуємо $W_{33}$ та з'єднуємо точки $W_{31}, W_{32}$ з найденою точкою прямими.

Крок 8. Побудова талієвої виточки по переду: посередині відрізку $W_1W_{22}$ ставиться точка $W_4$. Від точки $W_4$ униз опускається перпендикуляр рівний 8-10 см, чим більш Р.в тим довше витачка. Від точки $W_4$ відкладається вправо та вліво точки, на відстань $\frac{Р.в.сп.}{2}$ , та отримуємо $W_{41}$ і $W_{42}$ відповідно.

Крок 9. Оформлення виточки по лінії боку: від точки $W_{21}$ і $W_{22}$ проводиться вгору перпендикуляри рівні 1 см, отримуємо точки $W'_{21}$ і $W'_{22}$ відповідно, і з'єднується ці точки з точкою $H_2$.

Крок 10. Оформлення лінії бокового зрізу: із точки х (див. вираз (4.9)) проводиться перпендикуляр вправо на 0.5 см, а з точки $x_1$ (див. вираз (4.10)) – вліво, та через отримані перпендикуляри проводяться опуклі лінії.

$$x = \frac{W'_{21}\,H_2}{2} \tag{4.9}$$

$$x_1 = \frac{W'_{22}\,H_2}{2} \tag{4.10}$$

Крок 11. Оформлення зрізу лінії талії: точка $W'_{21}$ з'єднується з точкою $W_{32}$ плавною лінією, така ж операція повторюється з точками $W'_{22}$ та $W_{42}$.

### 4.2 Проектування моделі додатку

Матеріали про ідею та проектування додатка доповідалось на дев’ятнадцятої всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології» [19].

Сучасні системи поділяються на підсистеми для того, щоб збільшити гнучкість, масштабованість та полегшити тестування.

Проектна система побудована на основі архітектурного шаблону MVVM (рис. 4.1), тому розглянемо декомпозицію кожної частини на підсистеми.

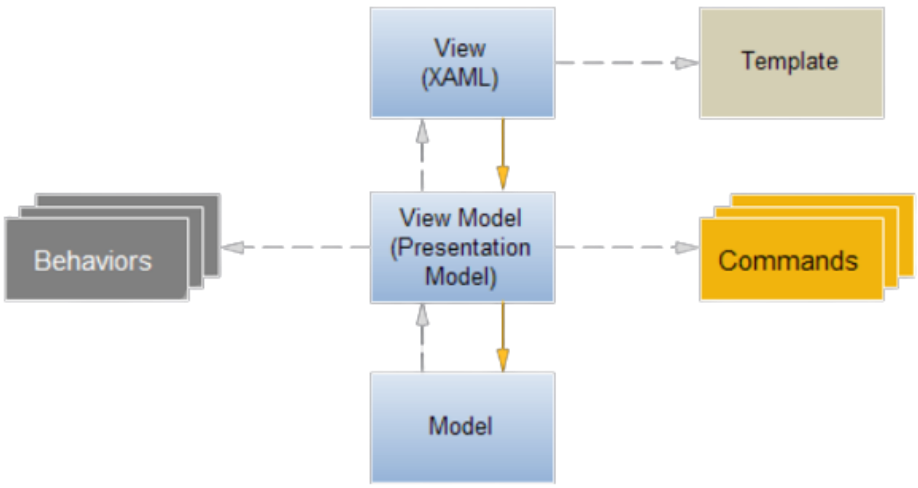


Рисунок 4.1 – Діаграма шаблону MVVM

Шаблон MVVM має три основних компонента:

1) модель, яка містить бізнес-логіку додатку;
2) уявлення користувацького інтерфейсу XAML;
3) модель-уявлення, що містить усю логіку побудови користувацького інтерфейсу та посилання на модель.

Розглянемо декомпозицію моделі.

Файли MeasurementsModel та BuiltSkirtModel описують необхідну бізнес-логіку.

Файли BuiltSkirtViewModel, MainWindowViewModel, Measurement-ViewModel зв’язують між собою модель та уявлення.

Файли MeasurementWindow, MainWindow, Construction описують уявлення вікон користувацького інтерфейсу.

Також використано механізм прив’язки (binding) для того, щоб ViewModel реалізувала інтерфейси INotifyPropertyChanged, ICommand.

Інтерфейс INotifyPropertyChanged реалізує систему сповіщень, яка активується коли значення властивостей змінюється. Це потрібно для моделі-уявлення, щоб механізм прив’язки користувацького інтерфейсу являвся динамічним.

Інтерфейс ICommand потрібен для прив’язки до певного XAML - елементу й визначає behavior елемента при певних діях.

Для більшого розуміння системи створена use-case діаграма (рис. 4.2), що дозволяє описати систему на концептуальному рівні, та діаграма з ієрархією вікон і елементів для користувача (рис.4.3).

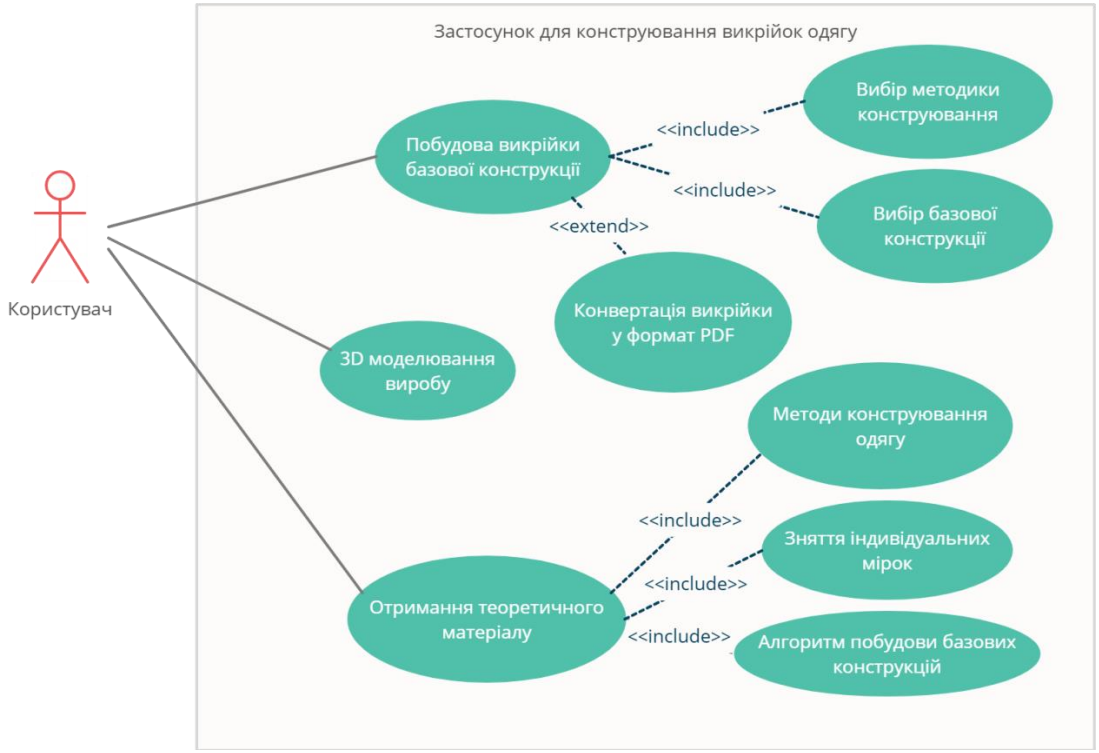


Рисунок 4.2 – Use-Case діаграма застосунку

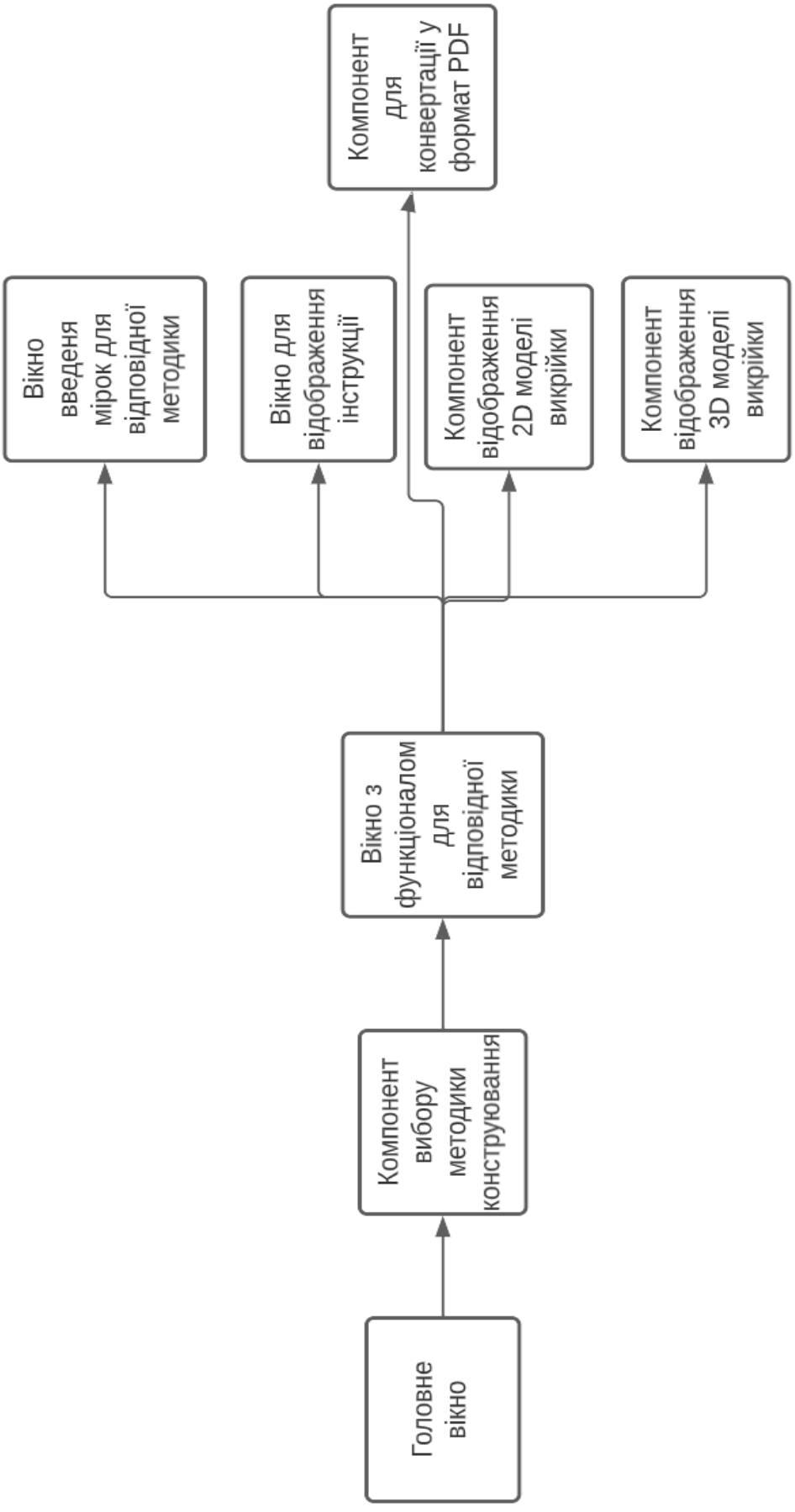


Рисунок 4.3 – Ієрархія вікон застосунку для користувача

# 5 РЕАЛІЗАЦІЯ СИСТЕМИ

## 5.1 Реалізація шаблону проектування MVVM

### 5.1.1. Вікно MainWindow

Метою MainWindow являється визначення обраної методики конструювання та показ певного функціоналу для певної методики.

У цьому випадку реалізується не усі шари шаблону (View – MainWindow.xaml, ViewModel – MainWindowViewModel.cs), так не маємо необхідності використовувати бізнес-логіку (рис. 5.1).

В моделі представлення використовується прив’язка подій, пов’язаних з закриттям форми та натисканням на кнопку відкриття нової форми.


Рисунок 5.1 – Реалізація MVVM для вікна MainWindow

### 5.1.2 Window Construction

У Window Construction реалізовано практично увесь функціонал, такий як побудову 3D моделі, конвертування викрійки у формат PDF та інше.

У цьому випадку повноцінно реалізується шаблон проектування (рис. 5.2), де:

1) View – у файлі Construction.xaml;
2) VievModel – у файлі BuiltSkirtViewModel.cs;
3) Model – у файлі BuiltSkirtModel.cs.


Рисунок 5.2 – Реалізація MVVM для вікна Construction

При загрузці форми відкривається вікно Measurements (дивись підрозділ 5.1.3). Вікно Window Construction являється родителем Measurements, що дозволяє реалізувати спілкування та передачу даних між ними.

Функція `public void DrawMeasurementsCRIGI(MeasurementsModel measurements, EnumTypeClothes typeClothes)` необхідна для виклику певного методу будування лекал, де measurements – мірки людини, typeClothes – певний тип одягу.

Для відкриття, перетворення та показу 3D моделі згідно з вимірами реалізовані функції `void Open3DModel(object sender, RoutedEventArgs e)` та `Model3DGroup Model3DSkirtCRIGI(Model3DGroup model)`, де перша функція шукає необхідну модель виробу, а Model3DSkirtCRIGI масштабує кожен елемент моделі згідно з Measurements.

Для того, щоб програма мала можливість конвертувати stackPanel (елемент інтерфейсу, де малюється 2D креслення) у PDF, створені такі функції:

1) `private void OnSaveXPSThenPreview(object sender, RoutedEventArgs e)`, де обирається місто збереження PDF – файлу, викликаються функції Export та Split та перетворює XPS в PDF;

2) `public void Export(string path, FrameworkElement surface)` – експортує викрійку з stackPanel та перетворює у формат XPS;

3) `public static PdfDocument Split(PdfDocument pdf, int number, bool isHorizontal)` – виконує дроблення викрійки на сторінки, відповідні формату А4.

Згідно з алгоритмом конструювання викрійки в підрозділі 4.1, функція StartBuildSkirt() розраховує та з’єднує основні точки між собою для утворення и показу креслення на екрані.

### 5.1.3 WindowMeasurements

У WindowMeasurements реалізована форма отримання мірок від користувача та їх обробка, при допомозі механізму прив’язки властивостей Binding у ViewModel. Перелік мірок можна подивитися у додатку А.

У цьому випадку теж повноцінно реалізується шаблон проектування, де (рис. 5.3):

1) View – у файлі Measurements.xaml;
2) VievModel – у файлі MeasurementsViewModel.cs;
3) Model – у файлі MeasurementsModel.cs.


Рисунок 5.3 – Реалізація MVVM для вікна MeasurementsWindow

Код застосунку можна переглянути у додатку Б. Загалом реалізація MVVM у застосунку наведено у додатку В.

## 5.2 Інструкція користувача

Розглянемо інструкцію для користувача.

Перше, що побачить користувач, після установки додатку – це головне вікно (рис. 5.4).

Далі користувачу надається можливість обрати по якій методиці будуть конструюватися викрійки (рис. 5.5).

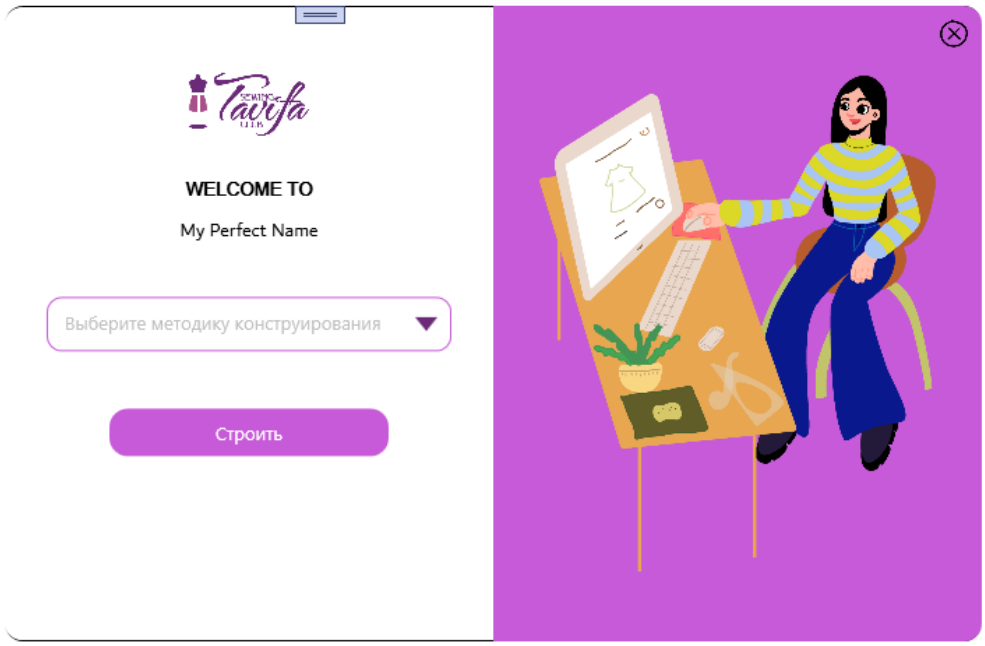


Рисунок 5.4 – Головне вікно

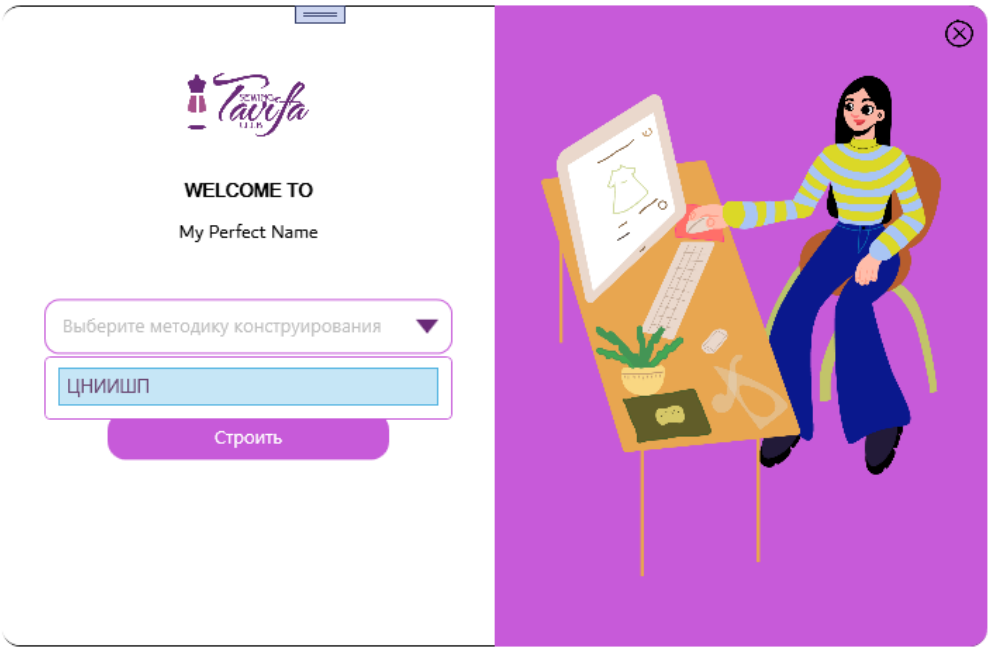


Рисунок 5.5 – Компонент вибору методики конструювання

Якщо користувач не обрав методику та натиснув на кнопку «Строить», то він побачить діалогове вікно с повідомленням – «Сперва выберите методику конструирования» (рис. 5.6).

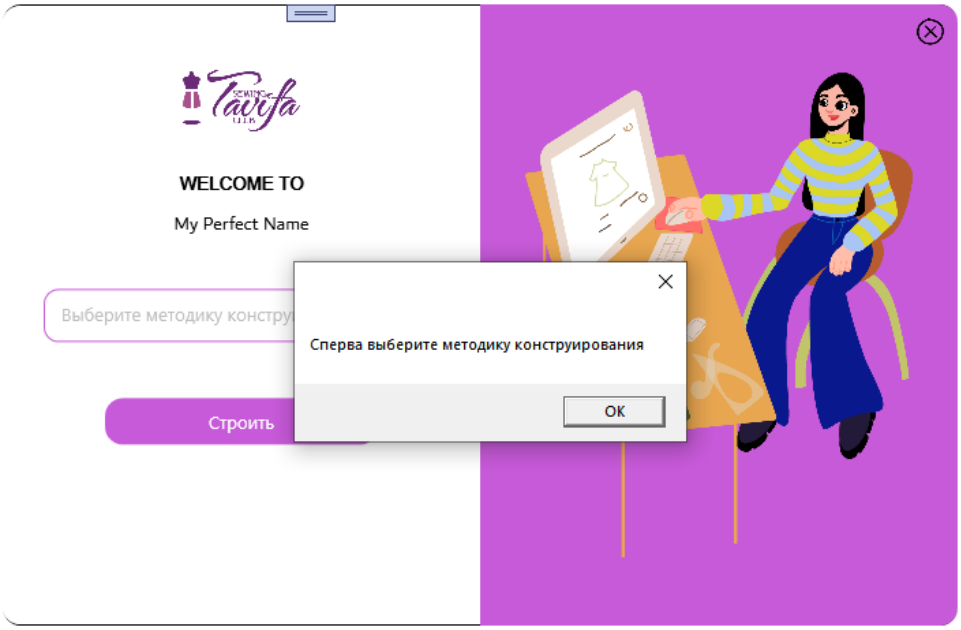


Рисунок 5.6 – Натискання кнопки без обраної методики

Якщо користувач обрав методику (рис. 5.7) та натиснув на кнопку «Строить», то з’явиться нове вікно, перед цим зачинивши головне вікно (рис.5.8).

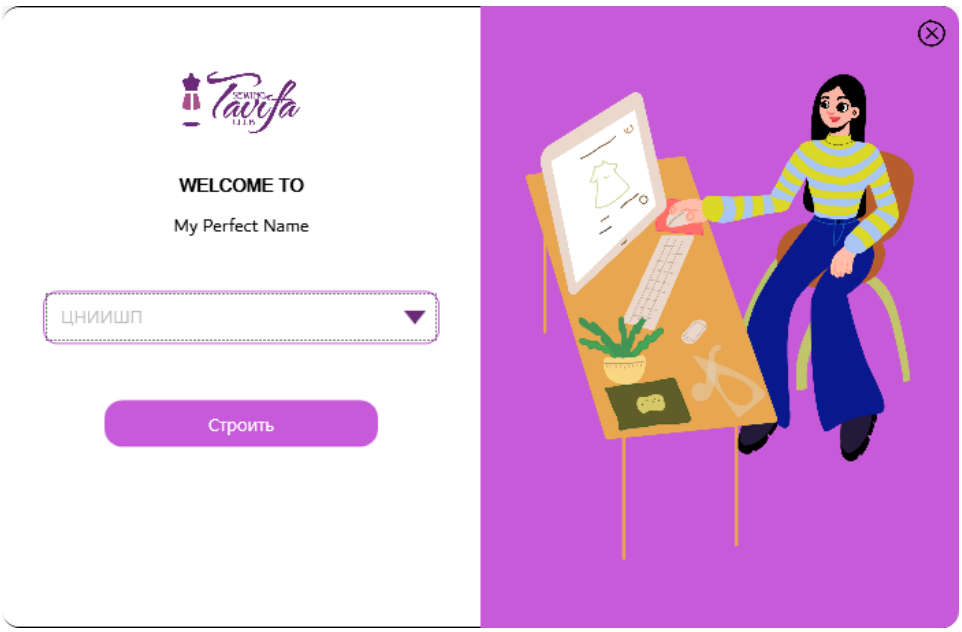


Рисунок 5.7 – Вигляд вікна після обрання методики

У новому вікні користувач може побачити дочірнє вікно, де він може обрати вид одягу та потім ввести свої персональні мірки (рис. 5.9). Мірки розташовані у такому порядку, як вони звичайно знімаються, для більшої зручності.

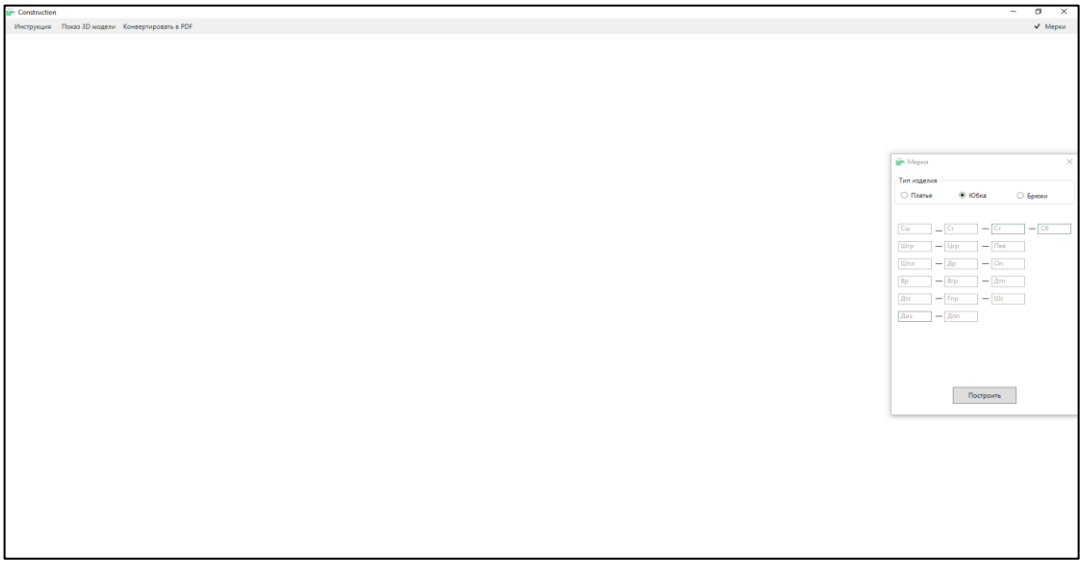

Рисунок 5.8 – Відкриття нового вікна після натискання кнопки «Строить»

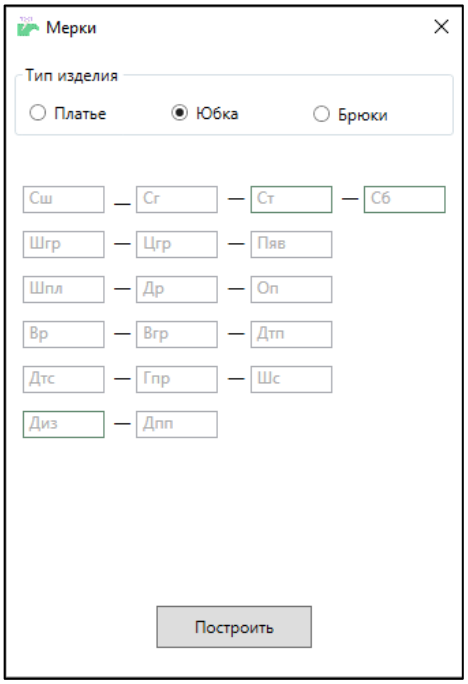


Рисунок 5.9 – Вікно мірок

Якщо користувач випадково закриє це вікно, натиснув на певну кнопку (див. рис. 5.10), то він зможе заново відкрити натиснув на кнопку «Мерки» зверху у меню (див. рис. 5.11).

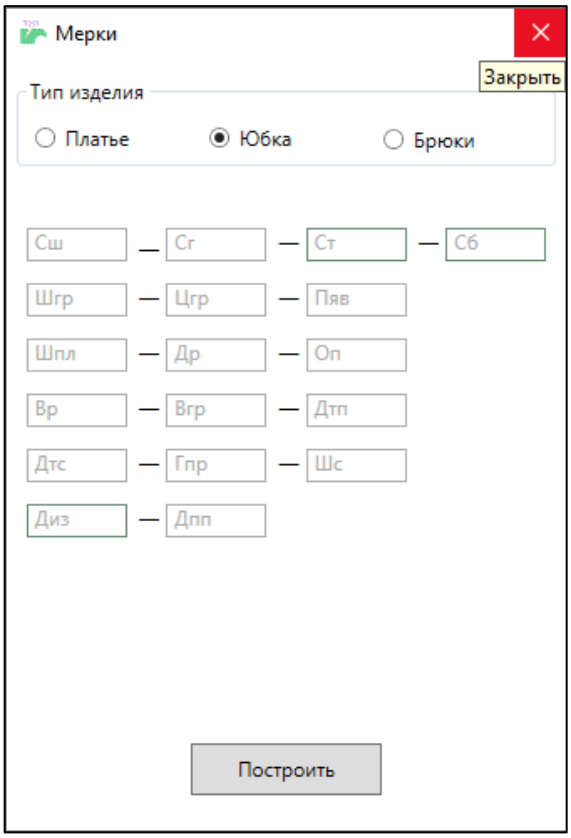


Рисунок 5.10 – Закриття вікна мірок

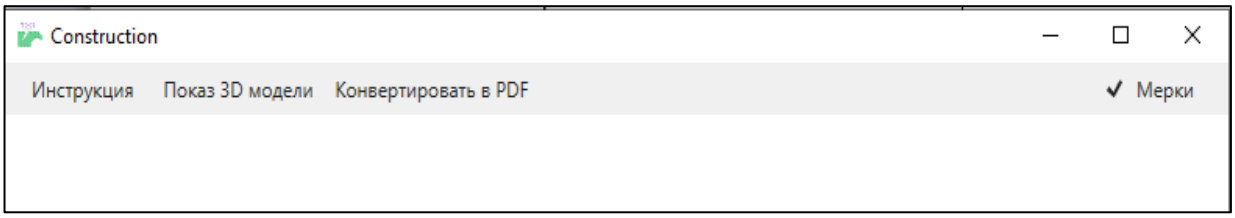


Рисунок 5.11 – Меню

Якщо користувачу потрібно роздрукувати 2D викрійку, то натиснув на кнопку «Конвертировать в PDF» (рис. 5.11), він отримує PDF-файл з роздробленою на А4 викрійку (рис. 5.12).

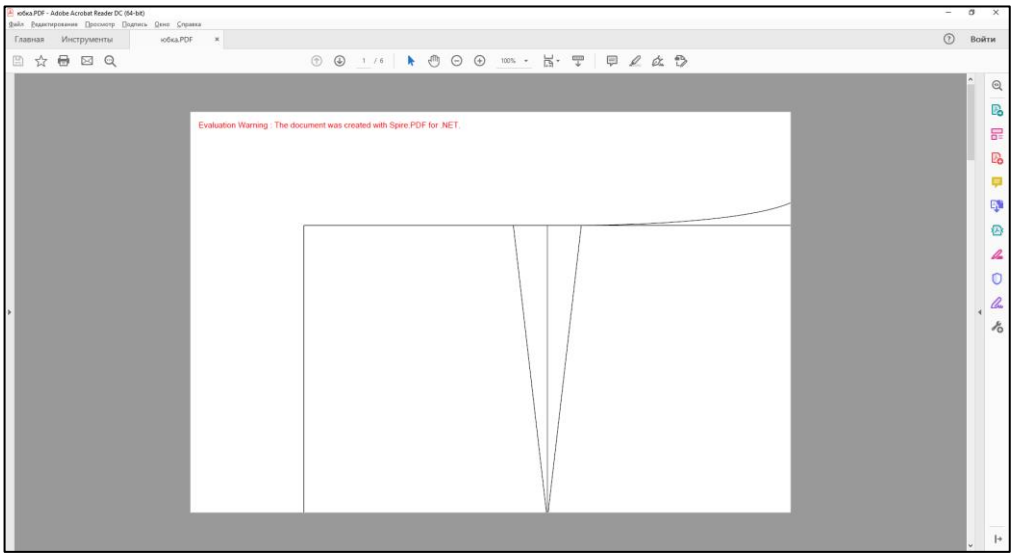


Рисунок 5.12 – PDF- файл з викрійкою, відкритий при допомозі AdobeAcrobat

Для отримання теоретичної інформації, треба натиснути на кнопку «Инструкция» (рис. 5.11).

Після того, як користувач вводить певні мірки та натискає кнопку «Построить» (див. рис. 5.13), будується викрійка обраного виду одягу (див. рис. 5.14). Викрійка будується в реальний масштаб, тобто у сантиметрах.

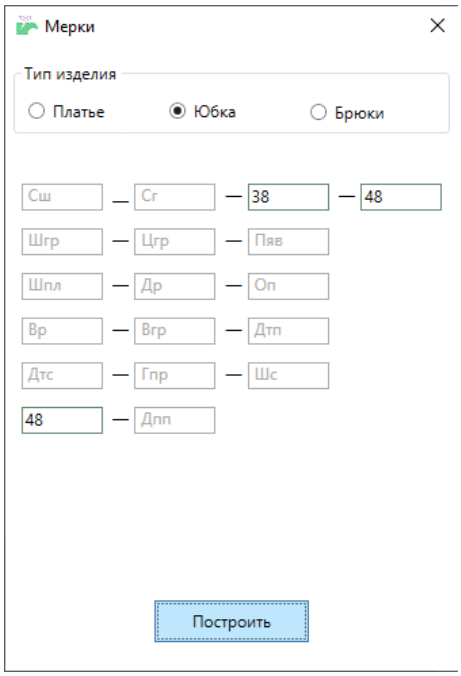


Рисунок 5.13 – Натискання на кнопку «Построить»

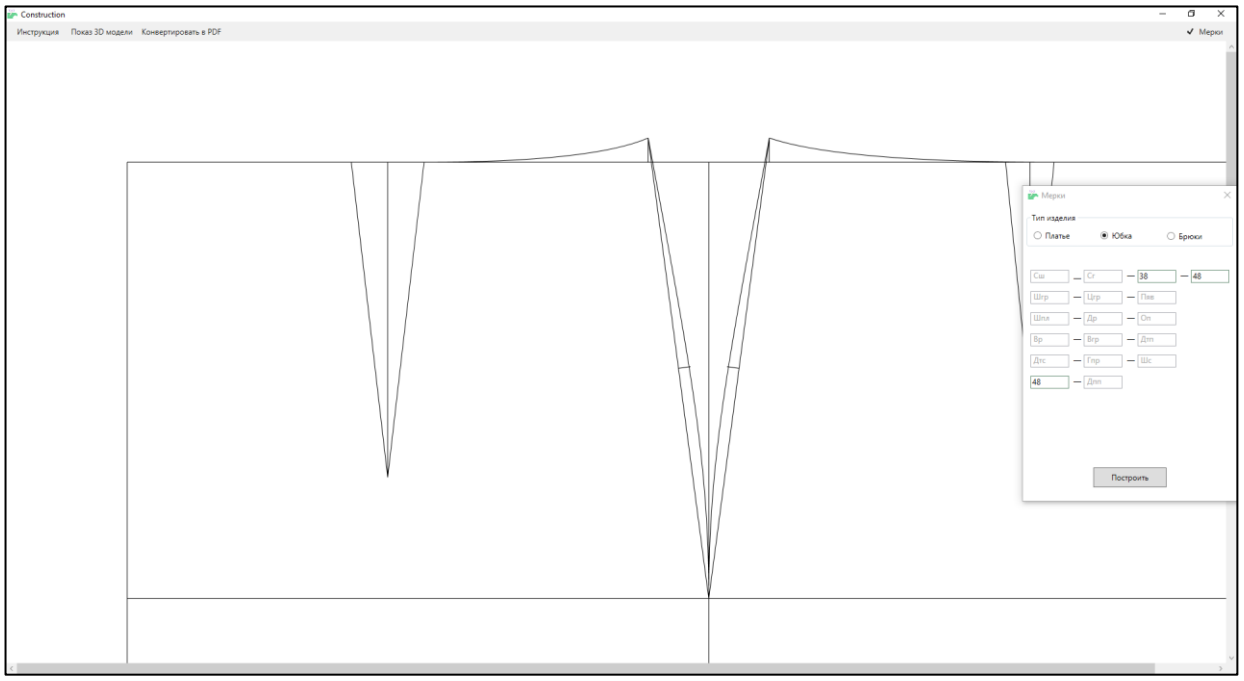


Рисунок 5.14 – Будування обраної викрійки

Якщо викрійка більш ніж дисплей екрану, то користувач може проскролити та переглянути увесь результат (рис. 5.15).

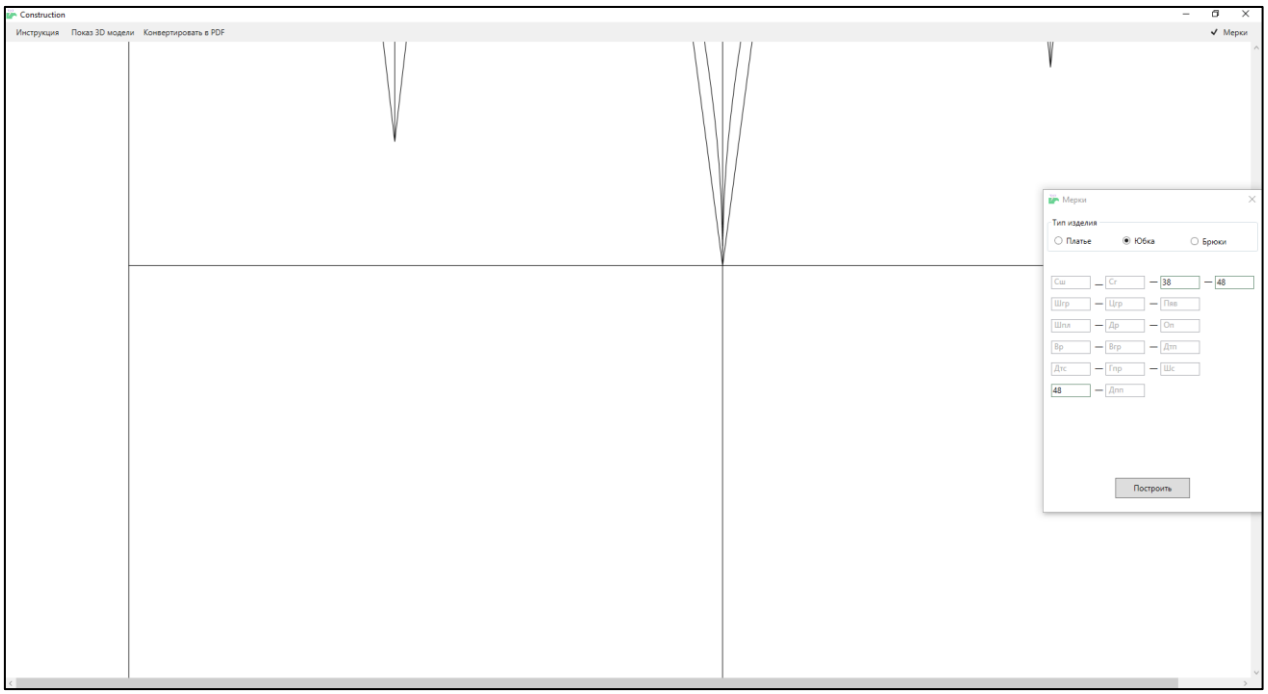


Рисунок 5.15 – Перегляд 2D викрійки

Також, користувач має можливість переглянути готову викрійку у вигляді 3D моделі (рис. 5.16).

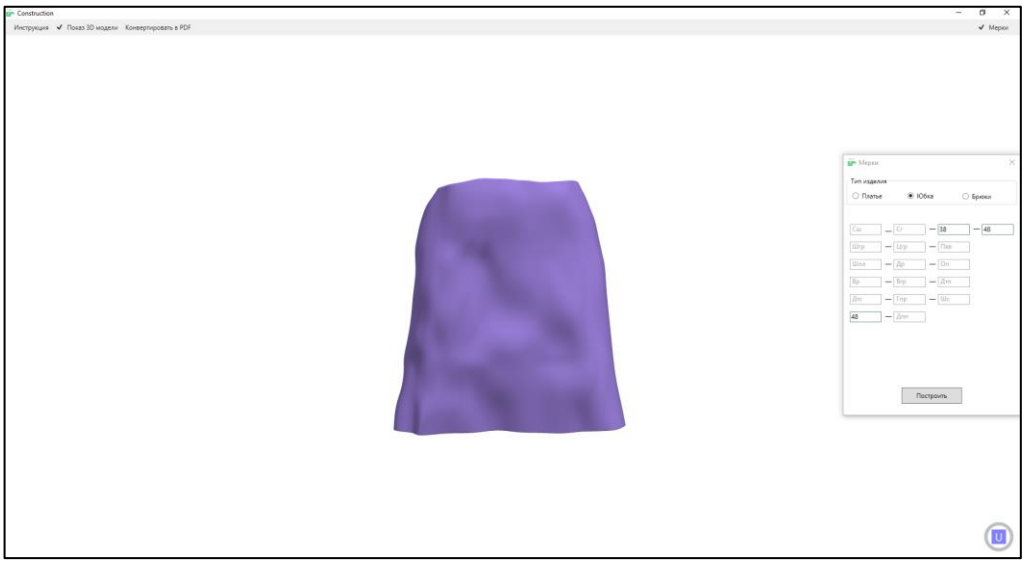

Рисунок 5.16 – 3D модель спідниці

Модель можна приближати та віддаляти (рис. 5.17), повернути у будь-який напрямок (рис. 5.18). Для різних величин мірок будуть відрізнятися пропорції моделі (рис. 5.19).

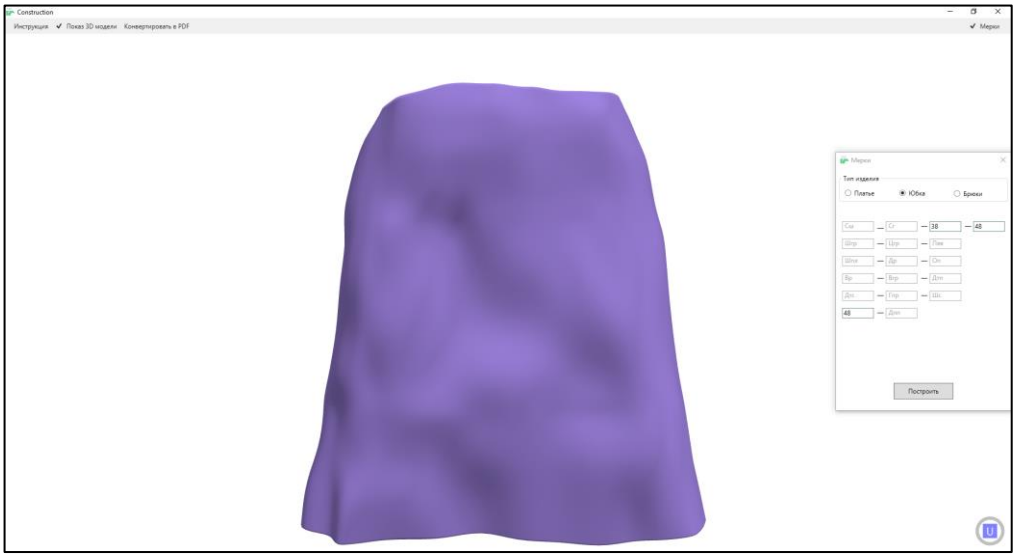

Рисунок 5.17 – Приближення моделі


Рисунок 5.18 – Поворот моделі


Рисунок 5.19 – Модель по викройці с іншими розмірними ознаками

# ВИСНОВКИ

В результаті виконання дипломного проекту проведено аналіз предметної області та виявлено основні проблеми, що турбують користувачів. Також, при огляду конкурентів виявлено, що вони більш орієнтуються на працю з великим виробництвом, ніж з маленькими ательє чи для індивідуального використання.

В наслідок чого, реалізовано десктопний застосунок для конструювання викрійок одягу, який вирішає описані вище проблеми.

Для реалізації системи використовуються надійні та сучасні засоби створення користувацького інтерфейсу разом з логікою на мові програмування C#, платформи .NET, та фреймворку WPF, з використанням шаблону проектування MVVM. 3D моделі створювалися в інструменті для роботи з 3D графікою Blender.

У застосунку користувач має можливість:

– отримати необхідну теоретичну інформацію стосовно конструювання одягу;

– побудувати викрійки на свої власні розмірні ознаки фігури тіла;

– подивитися модель викрійки виробу у 3D;

– обрати місце та зберегти викрійку у форматі PDF.

Розроблена програма має розвиток у майбутнього та може стати повноцінним додатком для розробки креслень різноманітного одягу, в якому матися самий необхідний функціонал та представлені методики конструювання на будь-який смак.

Застосунок розроблявся для використані в клубі крою та шиття «Tavifa» (див. додаток Г).

Матеріали роботи доповідались на дев’ятнадцятої всеукраїнській конференції студентів і молодих науковців «Інформатика, інформаційні системи та технології» (Одеса, 29 квітня 2022 р.).

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Google Trends [Електронний ресурс] // Режим доступу: https://trends.google.com/trends/explore?date=2015-01-01%202022-05-27&q=sewing%20classes%20near%20me
2. 60 Percent of People Haven't Learned This Household Skill [Електронний ресурс] // Режим доступу: https://www.marthastewart.com/1517398/60-percent-people-havent-learned-how-to-sew
3. Home-sewn clothes are making a comeback. But is it too late for dying fabric stores? [Електронний ресурс] // Режим доступу: https://www.sandiegouniontribune.com/business/retail/story/2020-01-31/homemade-clothes-are-making-a-comeback-can-slow-fashion-save-dying
4. Позитивний вплив рукоділля на жіноче здоров'я [Електронний ресурс] // Режим доступу: https://vbusk.com/ru/cikavo/pozytyvnyy-vplyv-rukodillia-na-zhinoche-zdorov-ia.html
5. САПР GRAFIS [Електронний ресурс] // Режим доступу: https://www.cadrus.ru/grafis/
6. САПР Julivi [Електронний ресурс] // Режим доступу: https://julivi.com/конструктор-одежды.html
7. САПР Assyst [Електронний ресурс] // Режим доступу: https://assyst-cis.com/
8. Cameo [Електронний ресурс] // Режим доступу: https://www.wildginger.com/default.htm
9. Характеристика методики ЦНИИШП [Електронний ресурс] // Режим доступу: http://wellconstruction.ru/konstr2/harakteristika-metodiki-tsniishp#:~:text=Речь%20пойдет%20о%20методике%20ЦНИИШП,фигур%2C%20закрепленных%20современными%20размерными%20стандартами.
10. Методики конструювання та моделювання одягу [Електронний ресурс] // Режим доступу: https://zakazlekal.ru/ru/polesnoe/15-construction.html

11. Design Patterns by Tutorials: MVVM [Електронний ресурс] // Режим доступу: https://www.raywenderlich.com/34-design-patterns-by-tutorials-mvvm
12. MOBILE & DESKTOP OPERATING SYSTEM MARKET SHARE STATS FOR 2022 [Електронний ресурс] // Режим доступу: https://earthweb.com /operating-system-market-share/
13. WHICH .NET FRAMEWORK FOR WINDOWS: UWP, WPF OR WINDOWS FORMS? [Електронний ресурс] // Режим доступу: https://www.itwriting.com/blog/10182-which-net-framework-for-windows-uwp-wpf-or-windows-forms.html
14. Кращі мови програмування [Електронний ресурс] // Режим доступу: https://blog.skillfactory.ru/samye-populyarnye-yazyki-programmirovaniya-2021-goda/
15. Welcome to Helix Toolkit’s documentation! [Електронний ресурс] // Режим доступу: http://docs.helix-toolkit.org/en/latest/index.html
16. Spire.PDF for .NET [Електронний ресурс] // Режим доступу: https://www.e-iceblue.com/Introduce/pdf-for-net-introduce.html#.YpPBQKhByUl
17. Introduction to Prism [Електронний ресурс] // Режим доступу: https://prismlibrary.com/docs/
18. Бланк А.Ф., Фомина З.М. Практическая книга по моделированию женской одежды / М.: Легпромбытиздат, 1995. – 255с.
19. Жмакіна А. С., Малахов Є. В. Десктопний застосунок для будування викрійок одягу / Інформатика, інформаційні системи та технології: тези доповідей дев’ятнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 29 квітня 2022 р. – Одеса, 2022. – с. 107-108.

# ДОДАТОК А

## Рекомендації щодо зняття мірок по методиці ЦНДІШП

Якщо мірка починається на літеру В, то це висота і мірка записується в повному розмірі.

Д – довжина

О – обхват

С – підлозі обхват (1\2)

Ц – центри (1\2)

Ш – ширини (1\2), виняток: Шпл – ширина плеча

Г – глибина

Мірки знімаються у кілька етапів:

1) $С_ш$ – $С_г$ – $С_т$ – $С_б$

$С_ш$ (напівобхват шиї / Half Neck (Hn) ) – СЛ позаду одним ребром накладається на 7-ий шийний хребець. Проходить на підставі шиї, замикаючись спереду на яремної впадині.

$С_г$ (напівобхват грудей / Half Bust (Hb) ) – СЛ проходить по нижчим кутам лопаток, горизонтально торкаючись нижчих кутів пахвових впадин, верхнім краєм спереду проходячи вищими точками грудних залоз.

$С_т$ (напівобхват талії / Half waist (Hw) ) – СЛ повинна проходити за найтоншим місцем тулуба і замикатися попереду паралельно підлозі.

$С_б$ (напівобхват стегон / Half hip (Hh) ) – СЛ ззаду лягає на виступаючі точки сідниць, охоплюючи стегна паралельно до підлоги. При яскраво вираженому животі мірка знімається з урахуванням виступу живота. Якщо тип фігури «груша», СЛ акуратно опускається вниз і фіксується велика величина, якщо така буде;

2) $Ш_{гр} – Ц_{гр} – П_{яв}$

$Ш_{гр}$ (ширина грудей / Bust Front (Bf) ) – відстань між двома перпендикулярами, зорово опущеними від точок зчленування руки з тулубом до лінії талії. СЛ проходить через виступаючі точки грудних залоз.

$Ц_{гр}$ (центр грудей / Bust Distance (Bd) ) – відстань між двома вищими точками грудних залоз.

$П_{яв}$ (положення яремної впадини / Neck Opening (On) ) – це відстань від площини паралельної підлоги, що виходить від яремної западини до площини перпендикулярної підлоги, що проходить через найвищі точки грудних залоз;

3) $Ш_{пл} – Д_{р} – О_{п}$

$Ш_{пл}$ (ширина плеча / Shoulder girth (Sg)) – СЛ лягає по центру плеча і заміряється відстань від основи шиї до плечової точки, яка знаходиться на перетині середньої лінії плечового ската, що поділяє плечовий суглоб навпіл.

$Д_{р}$ (довжина рукава / Sleeve Length (Sl)) – відстань від точки зчленування руки з тулубом до бажаної величини.

$О_{п}$ (обхват плеча / Bicep girth (Bg)) – СЛ лягає під руку до упору, рука знаходиться у вільному опущеному положенні і заміряється СЛ по руці паралельно до підлоги;

4) $В_{р}– В_{гр} – Д_{тп}$

$В_{р}$ (висота ростка / Center Front (Cf)) – відстань від лінії талії до найвищої точки підстави шиї. СЛ має лежати паралельно хребту.

$В_{гр}$ (висота грудей / Shoulder to Bust (Sb)) – відстань від найвищої точки основи шиї до найвищої точки грудних залоз. СЛ паралельна центру тулуба.

$Д_{тп}$ (довжина талії полочки / Shoulder to Waist (Sw)) – відстань від найвищої точки основи шиї до лінії талії, СЛ проходить через виступаючі точки грудних залоз. СЛ паралельна центру тулуба.

Для більш точного зняття цих мірок рекомендується знімати ці три мірки одночасно, а потім від отриманої величини $В_{гр}$ і $Д_{тп}$ віднімати $В_р$, тоді мірки виходять у чистому вигляді;

5) $Д_{тс} – Г_{пр} – Ш_с$

$Д_{тс}$ (довжина талії спинки / Center Back (Bc)) – вимірюється від сьомого шийного хребця до лінії талії. СЛ перпендикулярна до лінії талії.

$Г_{пр}$ (глибина пройми / armhole depth (Da)) – відстань від сьомого швейного хребця до перпендикуляра візуально проведеного від точки зчленування руки з тулубом до СЛ.

$Ш_с$ (ширина спинки / Chest Back (Bch)) – відстань між точками зчленування руки з тулубом. СЛ проходить через виступаючі точки лопаток і паралельно до підлоги;

6) $В_{пп} – Д_{из}$

$В_{пп}$ (висота плеча проекційна / Shoulder High (Hsh)) – це відстань від площини вихідної від вищої точки плечового ската і паралельної підлоги до площини перпендикулярної підлоги до нижньої точки плечового ската.

$Д_{из}$ (довжина виробу / Dress Length(Dl)) – мірка знімається по боці, від лінії талії до бажаної величини виробу.

# ДОДАТОК Б

## Вихідний код програми

### Код реалізації головного вікна

MainWindow.xaml

```
<Window x:Class="DiplomWork.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:DiplomWork"
    xmlns:viewModels="clr-namespace:DiplomWork.ViewModels"
    mc:Ignorable="d" WindowStartupLocation="CenterScreen"
    Title="MainWindow" Height="455" Width="700"
ResizeMode="NoResize"
    AllowsTransparency="True" Background="Transparent"
WindowStyle="None">
  <Window.DataContext>
    <viewModels:MainWindowViewModel/>
  </Window.DataContext>
  <Window.Resources>
    <Style x:Key="RoundButton" TargetType="Button">
      <Setter Property="Template">
        <Setter.Value>
          <ControlTemplate TargetType="Button">
            <Border CornerRadius="13"
                Background="{TemplateBinding Background}">
              <ContentPresenter
HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
```

```
                        VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
            </Border>
          </ControlTemplate>
        </Setter.Value>
      </Setter>
    </Style>
    <Style x:Key="CustomComboBox" TargetType="ComboBox">
      <Setter Property="Template">
        <Setter.Value>
          <ControlTemplate TargetType="ComboBox">
            <Border CornerRadius="13" BorderThickness="1"
                BorderBrush="{TemplateBinding BorderBrush}"
Background="#fff">
              <ContentPresenter
HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
                        VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
            </Border>
          </ControlTemplate>
        </Setter.Value>
      </Setter>
    </Style>
    <Style x:Key="ComboBoxInspectorStyle" TargetType="{x:Type
ComboBox}">
      <Setter Property="ToggleButton.FontSize" Value="13" />
      <Setter Property="ToggleButton.FontFamily" Value="Arial"
/>
      <Setter Property="ToggleButton.Foreground"
Value="#FF44184E" />
      <Setter Property="ToggleButton.Margin" Value="3" />
      <Setter Property="ToggleButton.Padding" Value="36" />
      <Setter Property="Template">
        <Setter.Value>
```

```
          <ControlTemplate TargetType="{x:Type ComboBox}">
            <Grid>
              <ToggleButton
x:Name="toggleButtonComboBoxInspectorStyle" Focusable="False"
IsChecked="{Binding Path=IsDropDownOpen, Mode=TwoWay,
RelativeSource={RelativeSource TemplatedParent}}"
ClickMode="Press" >
                <ToggleButton.Template>
                  <ControlTemplate TargetType="{x:Type
ToggleButton}">
                    <Grid>
                      <Grid.ColumnDefinitions>
                        <ColumnDefinition></ColumnDefinition>
                        <ColumnDefinition
Width="Auto"></ColumnDefinition>
                        <ColumnDefinition
Width="Auto"></ColumnDefinition>
                      </Grid.ColumnDefinitions>
                      <Rectangle
x:Name="rectangleComboBoxInspectorStyleFullContent"
Grid.Column="0" Grid.ColumnSpan="3" RadiusX="10" RadiusY="10"
StrokeThickness="1" Stroke="#C75AD9">
                        <Rectangle.Fill>
                          <LinearGradientBrush StartPoint="0, 0"
EndPoint="0 ,1">
                            <GradientStop Color="#FFF"
Offset="1"/>
                            <GradientStop Color="#FFF"/>
                          </LinearGradientBrush>
                        </Rectangle.Fill>
                      </Rectangle>
                      <Path
x:Name="pathArrowComboBoxInspectorStyle" Grid.Column="2"
Margin="10" HorizontalAlignment="Center"
```

```
VerticalAlignment="Center" Data="M 0 0 L 8 10 L 16 0 Z"
Fill="#6A2A78"/>
                    </Grid>
                    <ControlTemplate.Triggers>
                      <Trigger Property="IsChecked"
Value="True">
                        <Setter Property="Fill"
TargetName="rectangleComboBoxInspectorStyleFullContent"
Value="#FFF" />
                      </Trigger>
                    </ControlTemplate.Triggers>
                  </ControlTemplate>
                </ToggleButton.Template>
              </ToggleButton>
              <TextBox Text="{TemplateBinding Text}"
Foreground="Black" x:Name="textBoxComboBoxInspectorStyle"
Margin="{TemplateBinding Padding}" Style="{x:Null}"
HorizontalAlignment="Left" VerticalAlignment="Center"
Focusable="True" Background="Transparent" Visibility="Hidden"
IsReadOnly="{TemplateBinding IsReadOnly}">
                <TextBox.Template>
                  <ControlTemplate TargetType="{x:Type
TextBox}">
                    <Border
x:Name="borderTextBoxComboBoxInspectorStyle" Focusable="False"
Background="{TemplateBinding Background}" />
                  </ControlTemplate>
                </TextBox.Template>
              </TextBox>
              <Popup x:Name="popupComboBoxInspectorStyle"
Placement="Bottom" IsOpen="{TemplateBinding IsDropDownOpen}"
AllowsTransparency="True" Focusable="False"
PopupAnimation="Slide">
```

```
                <Grid x:Name="gridPopupComboBoxInspectorStyle"
SnapsToDevicePixels="True" MinWidth="{TemplateBinding
ActualWidth}" MaxHeight="{TemplateBinding MaxDropDownHeight}">
                  <Border
x:Name="borderPopupComboBoxInspectorStyle" Background="#FFF"
BorderThickness="1" BorderBrush="#C75AD9"/>
                  <ScrollViewer Margin="10, 10, 10, 10"
SnapsToDevicePixels="True">
                    <StackPanel IsItemsHost="True"
KeyboardNavigation.DirectionalNavigation="Contained" />
                  </ScrollViewer>
                </Grid>
              </Popup>
              <ContentPresenter
x:Name="contentPresenterComboBoxInspectorStyle" Grid.Column="0"
Margin="{TemplateBinding Padding}" IsHitTestVisible="False"
Content="{TemplateBinding SelectionBoxItem}"
ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"
ContentTemplateSelector="{TemplateBinding ItemTemplateSelector}"
VerticalAlignment="Center" HorizontalAlignment="Left" />
            </Grid>
            <ControlTemplate.Triggers>
              <Trigger Property="HasItems" Value="false">
                <Setter
TargetName="borderPopupComboBoxInspectorStyle"
Property="MinHeight" Value="95"/>
              </Trigger>
              <Trigger Property="IsGrouping" Value="true">
                <Setter Property="ScrollViewer.CanContentScroll"
Value="false"/>
              </Trigger>
              <Trigger SourceName="popupComboBoxInspectorStyle"
Property="Popup.AllowsTransparency" Value="true">
```

```
                <Setter
TargetName="borderPopupComboBoxInspectorStyle"
Property="CornerRadius" Value="4"/>
                <Setter
TargetName="borderPopupComboBoxInspectorStyle" Property="Margin"
Value="0,2,0,0"/>
              </Trigger>
              <Trigger Property="IsEditable" Value="true">
                <Setter Property="IsTabStop" Value="false"/>
                <Setter
TargetName="textBoxComboBoxInspectorStyle" Property="Visibility"
Value="Visible"/>
                <Setter
TargetName="contentPresenterComboBoxInspectorStyle"
Property="Visibility" Value="Hidden"/>
              </Trigger>
            </ControlTemplate.Triggers>
          </ControlTemplate>
        </Setter.Value>
      </Setter>
    </Style>
  </Window.Resources>
  <Grid>
    <Border Width="auto" Height="auto" BorderBrush="Black"
BorderThickness="0,1"
        CornerRadius="13,13,13,13">
      <Border.Background>
        <LinearGradientBrush>
          <GradientBrush.GradientStops>
            <GradientStopCollection>
              <GradientStop Color="#FFFFFF"
Offset="0.0"></GradientStop>
            </GradientStopCollection>
          </GradientBrush.GradientStops>
        </LinearGradientBrush>
```

```
      </Border.Background>
    </Border>
    <Border CornerRadius="0,13,13,0" Background="#C75AD9"
Margin="350,0,0,0"/>
    <Border Margin="132,48,482,362" Width="86" Height="45">
      <Border.Background>
        <ImageBrush
ImageSource="/Images/Logo_Small.png"></ImageBrush>
      </Border.Background>
    </Border>
    <Border Margin="383,50,33,50" Width="284" Height="355">
      <Border.Background>
        <ImageBrush
ImageSource="/Images/imageGirl.png"></ImageBrush>
      </Border.Background>
    </Border>
    <StackPanel Margin="8,124,358,127">
      <TextBlock TextAlignment="Center" FontSize="13"
FontFamily="Microsoft Sans Serif" Text="WELCOME TO"
FontWeight="Bold" />
      <TextBlock TextAlignment="Center" FontSize="14" Text="My
Perfect Name" Margin="10" FontFamily="Dubai"/>
      <ComboBox Height="39" Width="290" BorderBrush="#C75AD9"
Margin="0,26,0,26" Style="{StaticResource
ComboBoxInspectorStyle}" SelectionChanged="ComboBox_Selected">
        <ComboBoxItem Padding="5" Background="White"
Opacity="0.8" Content="ЦНИИШП"/>
      </ComboBox>
      <Button Margin="15" Width="200" Height="34"
Style="{StaticResource ResourceKey=RoundButton}"
Background="#C75AD9" FontSize="13" Foreground="#FFF"
Command="{Binding openCommand}">Строить</Button>
    </StackPanel>
    <StackPanel Width="20" Height="20" Margin="670,10,10,425">
      <TextBlock>
```

```
        <Hyperlink Command="{Binding closeCommand}"
TextDecorations="None">
          <Image Source="/Images/icons8-close-91.png" />
        </Hyperlink>
      </TextBlock>
    </StackPanel>
    <StackPanel Width="250" Height="35" Margin="38,213,412,207">
      <Label Name="TextComboBox" FontSize="13"
Foreground="#66000000" Content="Выберите методику
конструирования"/>
    </StackPanel>
  </Grid>
</Window>
```

MainWindow.xaml.cs

```
using DiplomWork.Enum;
using DiplomWork.ViewModels;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
namespace DiplomWork
{
  /// <summary>
  /// Interaction logic for MainWindow.xaml
  /// </summary>
  public partial class MainWindow : Window
  {
    Dictionary<string, EnumTypeConstructionMethods>
TypeConstructionMethods;
    public MainWindow()
    {
      InitializeComponent();
      Loaded += MainWindow_Loaded;
```

```
      TypeConstructionMethods = new Dictionary<string,
EnumTypeConstructionMethods>()
      {
        {"ЦНИИШП" , EnumTypeConstructionMethods.CRIGI}
      };
    }
    private void MainWindow_Loaded(object sender,
RoutedEventArgs e)
    {
      if(DataContext is IOpenCloseWindow vm)
      {
        vm.Close += () =>
         {
           this.Close();
         };
        vm.Open += ()=> OpenWindow();
      }
    }
    private void OpenWindow()
    {
      if (TextComboBox.Content.ToString() == "Выберите методику
конструирования")
      {
        MessageBox.Show("Сперва выберите методику
конструирования");
      }
      else
      {
        Construction openWindow = new
Construction(TypeConstructionMethods[TextComboBox.Content.ToStri
ng()]);
        openWindow.Show();
        this.Close();
      }
    }
```

```
    private void ComboBox_Selected(object sender,
RoutedEventArgs e)
    {
      ComboBox comboBox = (ComboBox)sender;
      TextComboBox.Content =
((ComboBoxItem)comboBox.SelectedItem).Content.ToString();
    }
  }
}
```

MainWindowViewModel.cs

```
using Prism.Commands;
using System;
namespace DiplomWork.ViewModels
{
  public class MainWindowViewModel:IOpenCloseWindow
  {
    private DelegateCommand _closeCommand;
    public DelegateCommand closeCommand => _closeCommand ??
(_closeCommand = new DelegateCommand(CloseWindow));
    public void CloseWindow()
    {
      Close?.Invoke();
    }
    public Action Close { get; set; }
    private DelegateCommand _openCommand;
    public DelegateCommand openCommand => _openCommand ??
(_openCommand = new DelegateCommand(OpenWindow));
    public void OpenWindow()
    {
      Open?.Invoke();
    }
```

```
    public Action Open { get; set; }
  }
  interface IOpenCloseWindow
  {
    Action Close { get; set; }
    Action Open { get; set; }
  }
}
```

**2 Код реалізації вікна для введення мірок**

MeasurementWindow.xaml

```
<Window x:Class="DiplomWork.MeasurementWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:DiplomWork"
    xmlns:i="http://schemas.microsoft.com/xaml/behaviors"
    xmlns:prism="http://prismlibrary.com/"
    mc:Ignorable="d" ResizeMode="NoResize"
    Title="Мерки" Height="500" Width="350">
  <Grid>
    <Grid Margin="0,0,0,392">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="3*"/>
        <ColumnDefinition Width="19*"/>
        <ColumnDefinition Width="281*"/>
        <ColumnDefinition Width="39*"/>
        <ColumnDefinition Width="8*"/>
      </Grid.ColumnDefinitions>
```

```
      <GroupBox x:Name="TypeClothesGroupBox" Header="Тип
изделия" Margin="2,10,5,4" Grid.ColumnSpan="4" Grid.Column="1">
        <Grid Height="46">
          <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
          </Grid.ColumnDefinitions>
          <RadioButton x:Name="DressRadio" Content="Платье"
Height="19" Width="60" Grid.Column="0"
HorizontalAlignment="Left" Margin="5,0,0,0"/>
          <RadioButton x:Name="SkirtRadio" Content="Юбка"
Height="19" Width="60" Grid.Column="1"
HorizontalAlignment="Left" Margin="5,0,0,0" IsChecked="True"/>
          <RadioButton x:Name="ThrousersRadio" Content="Брюки"
Height="19" Width="79" Grid.Column="2"
HorizontalAlignment="Left" Margin="5,14,0,14"/>
        </Grid>
      </GroupBox>
    </Grid>
    <Grid Margin="0,94,0,168">
      <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
        <ColumnDefinition/>
      </Grid.ColumnDefinitions>
      <Grid.RowDefinitions>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
      </Grid.RowDefinitions>
```

```
      <TextBox x:Name="TextboxHn" Height="20" Width="60"
Margin="0,0,0,0" HorizontalAlignment="Center"
TextWrapping="Wrap" Text="{Binding HalfNeck,
UpdateSourceTrigger=LostFocus}" Grid.Column="0" Grid.Row="0"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown">
      </TextBox>
      <TextBlock x:Name="Placeholder1" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Cm" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="0" Grid.Row="0">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxHn}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxHb" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding HalfBust,
UpdateSourceTrigger=LostFocus}"  Grid.Column="1" Grid.Row="0"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder2" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Cm" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="1" Grid.Row="0">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
```

```
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxHb}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxHw" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding HalfWaist,
UpdateSourceTrigger=LostFocus}" Grid.Column="2" Grid.Row="0"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"
BorderBrush="#FF648B72" Background="White"/>
      <TextBlock x:Name="Placeholder3" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Ст" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="2" Grid.Row="0">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxHw}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxHh" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding HalfHip,
UpdateSourceTrigger=LostFocus}" Grid.Column="3" Grid.Row="0"
```

```
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"
BorderBrush="#FF648B72"/>
      <TextBlock x:Name="Placeholder4" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Сб" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="3" Grid.Row="0">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxHh}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,0,0,0" VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Column="2"/>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,11,0,0" VerticalAlignment="Top" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2"/>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,0,0,0" VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Column="1"/>
      <TextBox x:Name="TextboxBf" Height="20" Width="60"
Margin="0,0,0,0" HorizontalAlignment="Center"
TextWrapping="Wrap" Text="{Binding BustFront,
```

```
UpdateSourceTrigger=LostFocus}" Grid.Column="0" Grid.Row="1"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder5" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Шгр" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="0" Grid.Row="1">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxBf}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxBd" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding
BustDistance, UpdateSourceTrigger=LostFocus}"  Grid.Column="1"
Grid.Row="1" VerticalAlignment="Center"
KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder6" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Цгр" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="1" Grid.Row="1">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxBd}" Value="">
```

```
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxOn" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding NeckOpening,
UpdateSourceTrigger=LostFocus}" Grid.Column="2" Grid.Row="1"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder7" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Пяв" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="2" Grid.Row="1">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxOn}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <Label Content="—" Margin="79,0,0,0"
VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Row="1"
HorizontalAlignment="Left" Width="12"/>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,0,0,0" VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
```

```
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Column="1"
Grid.Row="1"/>
      <TextBox x:Name="TextboxSg" Height="20" Width="60"
Margin="0,0,0,0" HorizontalAlignment="Center"
TextWrapping="Wrap" Text="{Binding ShoulderGirth,
UpdateSourceTrigger=LostFocus}" Grid.Column="0" Grid.Row="2"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder8" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Шпл" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="0" Grid.Row="2">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxSg}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxSl" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding
ShoulderLength, UpdateSourceTrigger=LostFocus}"  Grid.Column="1"
Grid.Row="2" VerticalAlignment="Center"
KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder9" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Др" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="1" Grid.Row="2">
        <TextBlock.Style>
```

```
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxSl}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxBg" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding BicepGirth,
UpdateSourceTrigger=LostFocus}" Grid.Column="2" Grid.Row="2"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder10" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Оп" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="2" Grid.Row="2">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxBg}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <Label Content="—" Margin="79,0,0,0"
VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
```

```
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Row="2"
HorizontalAlignment="Left" Width="12"/>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,0,0,0" VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Column="1"
Grid.Row="2"/>
      <TextBox x:Name="TextboxCf" Height="20" Width="60"
Margin="0,0,0,0" HorizontalAlignment="Center"
TextWrapping="Wrap" Text="{Binding CenterFront,
UpdateSourceTrigger=LostFocus}" Grid.Column="0" Grid.Row="3"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder11" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Bp" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="0" Grid.Row="3">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxCf}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxSb" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding
ShoulderToBust, UpdateSourceTrigger=LostFocus}"  Grid.Column="1"
Grid.Row="3" VerticalAlignment="Center"
KeyDown="Textbox_KeyDown"/>
```

```
      <TextBlock x:Name="Placeholder12" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Вгр" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="1" Grid.Row="3">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxSb}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxSw" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding
ShoulderToWaist, UpdateSourceTrigger=LostFocus}" Grid.Column="2"
Grid.Row="3" VerticalAlignment="Center"
KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder13" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Дтп" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="2" Grid.Row="3">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxSw}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
```

```
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <Label Content="—" Margin="79,0,0,0"
VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Row="3"
HorizontalAlignment="Left" Width="12"/>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,0,0,0" VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Column="1"
Grid.Row="3"/>
      <TextBox x:Name="TextboxBc" Height="20" Width="60"
Margin="0,0,0,0" HorizontalAlignment="Center"
TextWrapping="Wrap" Text="{Binding CenterBack,
UpdateSourceTrigger=LostFocus}" Grid.Column="0" Grid.Row="4"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder14" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Дтс" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="0" Grid.Row="4">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxBc}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
```

```
      </TextBlock>
      <TextBox x:Name="TextboxDa" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding
ArmholeDepth, UpdateSourceTrigger=LostFocus}"  Grid.Column="1"
Grid.Row="4" VerticalAlignment="Center"
KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder15" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Гпр" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="1" Grid.Row="4">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxDa}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxBch" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding ChestBack,
UpdateSourceTrigger=LostFocus}" Grid.Column="2" Grid.Row="4"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder16" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Шс" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="2" Grid.Row="4">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
```

```
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxBch}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <Label Content="—" Margin="79,0,0,0"
VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Row="4"
HorizontalAlignment="Left" Width="12"/>
      <Label Content="—" HorizontalAlignment="Left"
Margin="79,0,0,0" VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Column="1"
Grid.Row="4"/>
      <TextBox x:Name="TextboxDl" Height="20" Width="60"
Margin="0,0,0,0" HorizontalAlignment="Center"
TextWrapping="Wrap" Text="{Binding DressLength,
UpdateSourceTrigger=LostFocus}" Grid.Column="0" Grid.Row="5"
VerticalAlignment="Center" KeyDown="Textbox_KeyDown"
BorderBrush="#FF648B72"/>
      <TextBlock x:Name="Placeholder17" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Диз" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="0" Grid.Row="5">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
```

```
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxDl}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <TextBox x:Name="TextboxHsh" Height="20" Width="60"
Margin="0,0,0,0" TextWrapping="Wrap" Text="{Binding
ShoulderHigh, UpdateSourceTrigger=LostFocus}"  Grid.Column="1"
Grid.Row="5" VerticalAlignment="Center"
KeyDown="Textbox_KeyDown"/>
      <TextBlock x:Name="Placeholder18" IsHitTestVisible="False"
TextWrapping="Wrap" Text="Дпп" VerticalAlignment="Center"
Height="20" Width="55" Margin="5,0,0,0"
HorizontalAlignment="Center" Foreground="DarkGray"
Grid.Column="1" Grid.Row="5">
        <TextBlock.Style>
          <Style TargetType="{x:Type TextBlock}">
            <Setter Property="Visibility" Value="Collapsed"/>
            <Style.Triggers>
              <DataTrigger Binding="{Binding Text,
ElementName=TextboxHsh}" Value="">
                <Setter Property="Visibility" Value="Visible"/>
              </DataTrigger>
            </Style.Triggers>
          </Style>
        </TextBlock.Style>
      </TextBlock>
      <Label Content="—" Margin="79,0,0,0"
VerticalAlignment="Center" Height="20"
ScrollViewer.CanContentScroll="True" Padding="0,0,0,0"
UseLayoutRounding="False" Grid.ColumnSpan="2" Grid.Row="5"
HorizontalAlignment="Left" Width="12"/>
```

```
    </Grid>
    <Button x:Name="buttonBuild" Content="Построить"
HorizontalAlignment="Center" Margin="0,409,0,0"
VerticalAlignment="Top" Width="114" Height="31"
Command="{Binding openCommand}"/>
  </Grid>
</Window>
```

MeasurementWindow.xaml.cs

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using DiplomWork.ViewModels;
namespace DiplomWork
{
  /// <summary>
  /// Логика взаимодействия для MeasurementWindow.xaml
  /// </summary>
  public partial class MeasurementWindow : Window
  {
    public MeasurementWindow()
    {
      InitializeComponent();
      Loaded += MeasurementWindow_Loaded;
      DataContext = new MeasurementsViewModel();
    }
    private void MeasurementWindow_Loaded(object sender,
RoutedEventArgs e)
    {
      if (DataContext is IMeasurementsViewModel vm)
      {
        vm.Open += () => BuiltConstructionButton();
      }
      // event close window
```

```
      Closing += (s, e) =>
      {
         if (this.Owner is Construction)
         {
           Construction owner = (Construction)this.Owner;
           owner.MeasurementsMenuItem.IsChecked = false;
           e.Cancel = true;
           this.Visibility = Visibility.Hidden;
         }
      };
    }
    //event for the correct entry of measurements
    private void Textbox_KeyDown(object sender, KeyEventArgs e)
    {
      var ch = KeyEventUtility.GetCharFromKey(e.Key);

      if (ch == ','|| ch == '.')
      {
        TextBox txt = (TextBox)sender;
        if (txt.Text.Contains(".")|| txt.Text.Contains(",") ||
txt.Text=="")
        {
          e.Handled = true;
        }
        return;
      }
      if (!((e.Key >= Key.D0) && (e.Key <= Key.D9)))
      {
        e.Handled = true;
      }
    }
    // call method for building pattern
    private void BuiltConstructionButton()
    {
      if (this.Owner is Construction)
```

```
      {
        EnumTypeClothes typeCl = EnumTypeClothes.None;
        if (ThrousersRadio.IsChecked == true)
          typeCl = EnumTypeClothes.Throusers;
        else if (SkirtRadio.IsChecked == true)
        {
          typeCl = EnumTypeClothes.Skirt;
          if (TextboxHw.Text == "" || TextboxHh.Text == "" ||
TextboxDl.Text == "")
          {
            MessageBox.Show("Не заполнены необходимые поля (Ст,
Сб, Диз)");
            return;
          }
          if (Convert.ToDouble(TextboxHw.Text) < 21 ||
Convert.ToDouble(TextboxHh.Text) < 21 ||
Convert.ToDouble(TextboxDl.Text) < 21)
          {
            MessageBox.Show("Неправильно заполнены необходимые
поля (Ст, Сб, Диз)");
            return;
          }
          if (Convert.ToDouble(TextboxHw.Text) >=
Convert.ToDouble(TextboxHh.Text))
          {
            MessageBox.Show("Ст больше Сб");
            return;
          }
        }
        else if (DressRadio.IsChecked == true)
          typeCl = EnumTypeClothes.Dress;
        MeasurementsModel measur =
((MeasurementsViewModel)this.DataContext).GetAllMeasurements;
```

```
        ((Construction)this.Owner).DrawMeasurementsCRIGI(measur,
typeCl);
      }
    }
  }
}
```

MeasurentsWindowViewModel.cs

```
using Prism.Commands;
using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
namespace DiplomWork.ViewModels
{
  interface IMeasurementsViewModel
  {
    Action BuiltConstruction { get; set; }
    //Action<object, KeyEventArgs> TextBoxKeyDown { get; set; }
  }
  public class MeasurementsViewModel: INotifyPropertyChanged,
IMeasurementsViewModel
  {
    private MeasurementsModel _model;
    private DelegateCommand _builtPatternCommand;
    public DelegateCommand builtPatternCommand =>
_builtPatternCommand ?? (_builtPatternCommand = new
DelegateCommand(BuiltPatternWindow));
    public void BuiltPatternWindow()
    {
      BuiltConstruction?.Invoke();
    }
    public Action BuiltConstruction { get; set; }
    public MeasurementsViewModel()
    {
      _model = new MeasurementsModel();
```

```
    }
    public MeasurementsModel GetAllMeasurements
    {
      get
      {
        return _model;
      }
    }
    public string HalfNeck
    {
      get
      {
        return Convert.ToString(_model.Hn);
      }
      set
      {
        if (value != "")
          this._model.Hn = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Hn = null;
        OnPropertyChanged("HalfNeck");
      }
    }
    public string HalfBust
    {
      get
      {
        return Convert.ToString(_model.Hb);
      }
      set
      {
        if (value != "")
          this._model.Hb = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
```

```
        else
          this._model.Hb = null;
        OnPropertyChanged("HalfBust");
      }
    }
    public string HalfWaist
    {
      get
      {
        return Convert.ToString(_model.Hw);
      }
      set
      {
        if (value != "")
          this._model.Hw = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Hw = null;
        OnPropertyChanged("HalfWaist");
      }
    }
    public string HalfHip
    {
      get
      {
        return Convert.ToString(_model.Hh);
      }
      set
      {
        if (value != "")
          this._model.Hh = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Hh = null;
        OnPropertyChanged("HalfHip");
```

```
      }
    }
    public string BustFront
    {
      get
      {
        return Convert.ToString(_model.Bf);
      }
      set
      {
        if (value != "")
          this._model.Bf = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Bf = null;
        OnPropertyChanged("BustFront");
      }
    }
    public string BustDistance
    {
      get
      {
        return Convert.ToString(_model.Bd);
      }
      set
      {
        if (value != "")
          this._model.Bd = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Bd = null;
        OnPropertyChanged("BustDistance");
      }
    }
```

```
    public string NeckOpening
    {
      get
      {
        return Convert.ToString(_model.On);
      }
      set
      {
        if (value != "")
          this._model.On = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.On = null;
        OnPropertyChanged("NeckOpening");
      }
    }
    public string ShoulderGirth
    {
      get
      {
        return Convert.ToString(_model.Sg);
      }
      set
      {
        if (value != "")
          this._model.Sg = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Sg = null;
        OnPropertyChanged("ShoulderGirth");
      }
    }
    public string ShoulderLength
    {
      get
```

```
      {
        return Convert.ToString(_model.Sl);
      }
      set
      {
        if (value != "")
          this._model.Sl = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Sl = null;
        OnPropertyChanged("ShoulderLength");
      }
    }
    public string BicepGirth
    {
      get
      {
        return Convert.ToString(_model.Bg);
      }
      set
      {
        if (value != "")
          this._model.Bg = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Bg = null;
        OnPropertyChanged("BicepGirth");
      }
    }
    public string CenterFront
    {
      get
      {
        return Convert.ToString(_model.Cf);
      }
```

```
      set
      {
        if (value != "")
          this._model.Cf = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Cf = null;
        OnPropertyChanged("CenterFront");
      }
    }
    public string ShoulderToBust
    {
      get
      {
        return Convert.ToString(_model.Sb);
      }
      set
      {
        if (value != "")
          this._model.Sb = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Sb = null;
        OnPropertyChanged("ShoulderToBust");
      }
    }
    public string ShoulderToWaist
    {
      get
      {
        return Convert.ToString(_model.Sw);
      }
      set
      {
        if (value != "")
```

```
          this._model.Sw = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Sw = null;
        OnPropertyChanged("ShoulderToWaist");
      }
    }
    public string CenterBack
    {
      get
      {
        return Convert.ToString(_model.Bc);
      }
      set
      {
        if (value != "")
          this._model.Bc = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Bc = null;
        OnPropertyChanged("CenterBack");
      }
    }
    public string ArmholeDepth
    {
      get
      {
        return Convert.ToString(_model.Da);
      }
      set
      {
        if (value != "")
          this._model.Da = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
```

```
          this._model.Da = null;
        OnPropertyChanged("ArmholeDepth");
      }
    }
    public string ChestBack
    {
      get
      {
        return Convert.ToString(_model.Bch);
      }
      set
      {
        if (value != "")
          this._model.Bch = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Bch = null;
        OnPropertyChanged("ChestBack");
      }
    }
    public string ShoulderHigh
    {
      get
      {
        return Convert.ToString(_model.Hsh);
      }
      set
      {
        if (value != "")
          this._model.Hsh = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Hsh = null;
        OnPropertyChanged("ShoulderHigh");
      }}
```

```
    public string DressLength
    {
      get
      {
        return Convert.ToString(_model.Dl);
      }
      set
      {
        if (value != "")
          this._model.Dl = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
        else
          this._model.Dl = null;
        OnPropertyChanged("DressLength");
      }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged([CallerMemberName] string prop
= "")
    {
      if (PropertyChanged != null)
        PropertyChanged(this, new
PropertyChangedEventArgs(prop));
    }
  }
}
```

MeasurementsModel.cs

```
 namespace DiplomWork
 {
   public class MeasurementsModel
   {
     // half neck
     public double? Hn { get; set; }
     // half bust
```

```
        public double? Hb { get; set; }
        // half waist
        public double? Hw { get; set; }
        // half hip
        public double? Hh { get; set; }
        // bust front
        public double? Bf { get; set; }
        // bust distance
        public double? Bd { get; set; }
        // neck opening
        public double? On { get; set; }
        // shoulder girth
        public double? Sg { get; set; }
        // shoulder length
        public double? Sl { get; set; }
        // bicep girth
        public double? Bg { get; set; }
        // center front
        public double? Cf { get; set; }
        // shoulder to bust
        public double? Sb { get; set; }
        // shoulder to waist
        public double? Sw { get; set; }
        // center back
        public double? Bc { get; set; }
        // armhole depth
        public double? Da { get; set; }
        // chest back
        public double? Bch { get; set; }
        // shoulder high
        public double? Hsh { get; set; }
        // dress length
        public double? Dl { get; set; }
    }
}
```

## 3 Код реалізації вікна з основним функціоналом

Construction.xaml

```
<Window x:Name="ConstructionWindow"
x:Class="DiplomWork.Construction"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:h="http://helix-toolkit.org/wpf"
    xmlns:local="clr-namespace:DiplomWork"
    Loaded="OnLoad"
    mc:Ignorable="d" WindowState="Maximized"
    Title="Construction" Height="450" Width="800">
  <Grid>
    <Menu HorizontalAlignment="Stretch" Height="30"
VerticalAlignment="Top" Padding="5">
      <Menu.ItemsPanel>
        <ItemsPanelTemplate>
          <DockPanel HorizontalAlignment="Stretch"/>
        </ItemsPanelTemplate>
      </Menu.ItemsPanel>
      <MenuItem Header="Инструкция" Margin="5,0,0,0">
        <MenuItem Header="Инструкция пользователя"
Click="ConvertWordDocToXPSDoc"/>
        <MenuItem Header="Построение изделий">
          <MenuItem Header="Юбка"
Click="ConvertWordDocToXPSDoc"/>
        </MenuItem>
      <MenuItem Header="Снятие мерок"
Click="ConvertWordDocToXPSDoc"/>
```

```
      </MenuItem>
      <MenuItem x:Name="Model3DMenuItem" Header="Показ 3D
модели" Margin="5,0,0,0" Command="{Binding Open3DModel}"
IsCheckable="True" IsChecked="False"></MenuItem>
      <MenuItem Header="Конвертировать в PDF"
Click="OnSaveXPSThenPreview" Height="20"
VerticalAlignment="Top"/>
      <MenuItem x:Name="MeasurementsMenuItem" Header="Мерки"
HorizontalAlignment="Right" IsCheckable="True" Margin="0,0,10,0"
Click="ClickMeasurementsMenuItem" IsChecked="True">
        <!--<MenuItem.Icon >
          <Image Source="/Images/ruler.png" />
        </MenuItem.Icon>-->
      </MenuItem>
    </Menu>
    <ScrollViewer Name="Show2DModel"
VerticalScrollBarVisibility="Auto"
HorizontalScrollBarVisibility="Auto" Margin="0,30,0,0">
      <StackPanel Name="PanelForFigure">
        <StackPanel.RenderTransform>
          <ScaleTransform CenterX="0" CenterY="0" ScaleX="1"
ScaleY="1" />
        </StackPanel.RenderTransform>
      </StackPanel>
    </ScrollViewer>
    <h:HelixViewport3D Name="Show3DModel" Margin="0,30,0,0"
ZoomExtentsWhenLoaded="True" CameraRotationMode="Trackball"
Visibility="Hidden" Background="White">
      <h:HelixViewport3D.Camera>
        <PerspectiveCamera Position="0,1,8" LookDirection="0,0,-
6" UpDirection="0,1,0" NearPlaneDistance="0.01"
FarPlaneDistance="1000" FieldOfView="100"/>
      </h:HelixViewport3D.Camera>
      <h:DefaultLights/>
      <ModelVisual3D x:Name="TestModel"/>
```

```
    </h:HelixViewport3D>
  </Grid>
</Window>
```

Construction.xaml.cs

```
using DiplomWork.Enum;
using DiplomWork.ViewModels;
using System.Windows;
using System.Windows.Media;
using System.Windows.Xps.Packaging;
using System.Windows.Xps;
using System.IO.Packaging;
using System;
using System.IO;
using System.Windows.Controls;
using Microsoft.Win32;
using Spire.Pdf;
using Spire.Pdf.Graphics;
using HelixToolkit.Wpf;
using System.Windows.Media.Media3D;
using System.Windows.Interop;
using System.Collections.Generic;
namespace DiplomWork
{
  /// <summary>
  /// Логика взаимодействия для Construction.xaml
  /// </summary>
  public partial class Construction : Window
  {
    const double constHh = 92;
    const double constLength = 48;
    EnumTypeConstructionMethods typeConstructionMethods;
    Window neededTypeWindowMeasure;
    EnumTypeClothes typeClothesPattern=EnumTypeClothes.None;
```

```
    Dictionary<string, string> PathDocuments = new
Dictionary<string, string>
    {
      {"Инструкция пользователя", "Documents/Instructions.xps"},
      {"Юбка", "Documents/Skirts.xps" },
      {"Снятие мерок", "Documents/Measurements.xps" }
    };
    public Construction(EnumTypeConstructionMethods
_typeConstructionMethods)
    {
      InitializeComponent();
      typeConstructionMethods = _typeConstructionMethods;
    }
    void OnLoad(object sender, RoutedEventArgs e)
    {
      FindNeedConstruction();
      OpenMeasurement();
      if (DataContext is IBuiltSkirtViewModel vm)
      {
        vm.Open3DModel += () => Open3DModel();
      }
    }
    private void ClickMeasurementsMenuItem(object sender,
RoutedEventArgs e)
    {
      if(MeasurementsMenuItem.IsChecked)
      {
        OpenMeasurement();
      }
      else
      {
        MeasurementsMenuItem.IsChecked = false;
        foreach (Window w in this.OwnedWindows)
          w.Visibility = Visibility.Hidden;
      }}
```

```
    public void DrawMeasurementsCRIGI(MeasurementsModel
measurements, EnumTypeClothes typeClothes)
    {
      PanelForFigure.Children.Clear();
      System.Windows.Shapes.Path p = new
System.Windows.Shapes.Path();
      typeClothesPattern = typeClothes;
      switch (typeClothes)
      {
        case EnumTypeClothes.Skirt:
          DataContext = new BuiltSkirtViewModel();
          ((BuiltSkirtViewModel)DataContext).Measurements =
measurements;
          p.Data =
((BuiltSkirtViewModel)DataContext).CoordinatGeom;
          p.Stroke = Brushes.Black;
          PanelForFigure.Children.Add(p);
          Show3DModel.Camera.Position = new Point3D(0, 1, 8);
          Show3DModel.Camera.LookDirection = new Vector3D(0, 0,
-6);
          var objReader = new ObjReader();
          switch (typeConstructionMethods)
          {
            case EnumTypeConstructionMethods.CRIGI:
              try
              {
                TestModel.Content =
Model3DSkirtCRIGI(objReader.Read("3dmodel/testskirt.obj"));
              }
              catch(Exception ex)
              {
                MessageBox.Show(ex.Message, "Error!");
              }
              break;
            default:
```

```
                MessageBox.Show("Находится в разработке");
                break;
            }
            break;
          default:
            MessageBox.Show("Находится в разработке");
            break;
        }
      }
      private void OpenMeasurement()
      {
        var measureWindow = neededTypeWindowMeasure;
        measureWindow.ShowInTaskbar = false;
        measureWindow.Owner = this;
        measureWindow.Left = this.Width + this.Left -
measureWindow.Width;
        measureWindow.Top = this.Height / 2 - measureWindow.Height
/ 2;
        measureWindow.Show();
      }
      private void FindNeedConstruction()
      {
        switch(typeConstructionMethods)
        {
          case EnumTypeConstructionMethods.CRIGI:
            neededTypeWindowMeasure = new MeasurementWindow();
            break;
          default:
            MessageBox.Show("Находится в разработке");
            break;
        }
      }
      public void Export(string path, FrameworkElement surface)
      {
```

```
      if (path == null) return;
      // Save current canvas transorm
      Transform transform = surface.LayoutTransform;
      // Temporarily reset the layout transform before saving
      surface.LayoutTransform = null;
      // Get the size of the canvas
      Size size = new Size(surface.ActualWidth,
surface.ActualHeight);
      // Measure and arrange elements
      surface.Measure(size);
      surface.Arrange(new Rect(size));
      // Open new package
      Package package = Package.Open(path, FileMode.Create);
      // Create new xps document based on the package opened
      XpsDocument doc = new XpsDocument(package);
      // Create an instance of XpsDocumentWriter for the
document
      XpsDocumentWriter writer =
XpsDocument.CreateXpsDocumentWriter(doc);
      // Write the canvas (as Visual) to the document
      writer.Write(surface);
      // Close document
      doc.Close();
      // Close package
      package.Close();
      // Restore previously saved layout
      surface.LayoutTransform = transform;
    }
    private void Open3DModel()
    {
      try
      {
        if (Model3DMenuItem.IsChecked)
        {
          if (typeClothesPattern == EnumTypeClothes.None)
```

```
            {
              Model3DMenuItem.IsChecked = false;
              MessageBox.Show("Постройте сначала выкройку.");
            }
            else
            {
              Show3DModel.Camera.Position = new Point3D(0, 1, 8);
              Show3DModel.Camera.LookDirection = new Vector3D(0,
0, -6);
              Show3DModel.Visibility = Visibility.Visible;
              var objReader = new ObjReader();
              switch (typeConstructionMethods)
              {
                case EnumTypeConstructionMethods.CRIGI:
                  switch (typeClothesPattern)
                  {
                    case EnumTypeClothes.Skirt:
                      TestModel.Content =
Model3DSkirtCRIGI(objReader.Read("3dmodel/testskirt.obj"));
                      break;
                    default:
                      MessageBox.Show("Находится в разработке");
                      break;
                  }
                  break;
                default:
                  MessageBox.Show("Находится в разработке");
                  break;
              }
            }
          }
          else
          {
            Show3DModel.Visibility = Visibility.Hidden;
          }
```

```
      }
      catch (Exception ex)
      {
        MessageBox.Show(ex.Message, "Error!");
      }
    }
    private void OnSaveXPSThenPreview(object sender,
RoutedEventArgs e)
    {
      try
      {
        SaveFileDialog openFileDialog = new SaveFileDialog();
        if (openFileDialog.ShowDialog() == true)
        {
          string FilePath =
System.IO.Path.GetDirectoryName(openFileDialog.FileName) + @"\"+
System.IO.Path.GetFileNameWithoutExtension(openFileDialog.FileNa
me);
          string xpsName = "xpsMarketFile.XPS";
          if
(System.IO.Path.GetFileNameWithoutExtension(openFileDialog.FileN
ame) == "")
            FilePath += @"\fileMaket";
          //Save it as XPS file format
          Export(xpsName, PanelForFigure);
          using (PdfDocument doc = new PdfDocument())
          {
            doc.LoadFromFile(xpsName, Spire.Pdf.FileFormat.XPS);
            //Number of pieces in the horizontal direction
            int horizontalNumber =
(int)Math.Ceiling(doc.Pages[0].ActualSize.Width /
iTextSharp.text.PageSize.A4.Width);
            //Number of pieces in the vertical direction
```

```
            int verticalNumber =
(int)Math.Ceiling(doc.Pages[0].ActualSize.Height /
iTextSharp.text.PageSize.A4.Height);
            //!!!Note, horizontalNumber and verticalNumber must
be > 0
            //Split the page horizontally into equal-sized pages
            PdfDocument temPdf = Split(doc, horizontalNumber,
true);
            //Split the page vertically into equal-sized pages
            PdfDocument newPdf = Split(temPdf, verticalNumber,
false);
            //Save the Pdf document
            newPdf.SaveToFile(FilePath + ".PDF",
Spire.Pdf.FileFormat.PDF);
            newPdf.Close();
            doc.Close();
          }
        }
        else
        {
          MessageBox.Show("Выберите место для сохранения pdf");
        }
      }
      catch(Exception ex)
      {
        MessageBox.Show(ex.Message, "Error!");
      }
    }
    public static PdfDocument Split(PdfDocument pdf, int number,
bool isHorizontal)
    {
      //Create a new Pdf
      PdfDocument newPdf = new PdfDocument();
      //Remove all the margins
      newPdf.PageSettings.Margins.All = 0;
```

```
      //Spilt pages
      foreach (PdfPageBase page in pdf.Pages)
      {
        //Set the page size of new Pdf
        if (isHorizontal)
        {
          newPdf.PageSettings.Width = page.Size.Width;
          newPdf.PageSettings.Height = page.Size.Height /
number;
        }
        else
        {
          newPdf.PageSettings.Width = page.Size.Width / number;
          newPdf.PageSettings.Height = page.Size.Height;
        }
        //Add a new page
        PdfPageBase newPage = newPdf.Pages.Add();
        PdfTextLayout format = new PdfTextLayout();
        format.Break = PdfLayoutBreakType.FitPage;
        format.Layout = PdfLayoutType.Paginate;
        //Draw the page in the new page
        page.CreateTemplate().Draw(newPage, new
System.Drawing.PointF(0, 0), format);
      }
      return newPdf;
    }
    public Model3DGroup Model3DSkirtCRIGI(Model3DGroup model)
    {
      var newGroupModel = new Model3DGroup();
      // skirt - 0, молния - 1, бегунок - 2
      double x, y, z;
      var constA = 3 * constHh / (8 * Math.PI + 4);
```

```
      var Hh = ((BuiltSkirtViewModel)DataContext).
Measurements.Hh*2;
      var a = 3 * (double)Hh / (8 * Math.PI + 4);
      x = a / constA;
      z = x;
      y =
(double)((BuiltSkirtViewModel)DataContext).Measurements.Dl/const
Length;
      var skirt = model.Children[0] as GeometryModel3D;
      skirt.Transform = new ScaleTransform3D(x, y, z);
      skirt.Material = new DiffuseMaterial(new
SolidColorBrush(Color.FromRgb(171, 141, 242)));
      newGroupModel.Children.Add(skirt);
      var zipper = model.Children[1] as GeometryModel3D;
      zipper.Transform = new ScaleTransform3D(x, y, z);
      zipper.Material = new DiffuseMaterial(new
SolidColorBrush(Color.FromRgb(94, 77, 135)));
      newGroupModel.Children.Add(zipper);
      var slider = model.Children[2] as GeometryModel3D;
      slider.Transform = new ScaleTransform3D(x, y, z);
      slider.Material = new DiffuseMaterial(new
SolidColorBrush(Color.FromRgb(0, 0, 0)));
      newGroupModel.Children.Add(slider);
      return newGroupModel;
    }
    /// <summary>
    /// Transforms device independent units (1/96 of an inch)
    /// to pixels
    /// </summary>
    /// <param name="visual">a visual object</param>
    /// <param name="unitX">a device independent unit value
X</param>
    /// <param name="unitY">a device independent unit value
Y</param>
```

```
    /// <param name="pixelX">returns the X value in
pixels</param>
    /// <param name="pixelY">returns the Y value in
pixels</param>
    public void TransformToPixels(Visual visual,
                double unitX,
                double unitY,
                out double pixelX,
                out double pixelY)
    {
      Matrix matrix;
      var source = PresentationSource.FromVisual(visual);
      if (source != null)
      {
        matrix = source.CompositionTarget.TransformToDevice;
      }
      else
      {
        using (var src = new HwndSource(new
HwndSourceParameters()))
        {
          matrix = src.CompositionTarget.TransformToDevice;
        }
      }
      pixelX = matrix.M11 * unitX;
      pixelY = matrix.M22 * unitY;
    }
    //show instructions
    private void ShowXPSDoc(object sender, RoutedEventArgs e)
    {
      try
      {
        string nameItem =
Convert.ToString(((MenuItem)sender).Header);
        string filename = PathDocuments[nameItem];
```

```
        DocumentViewer viewer = new DocumentViewer();
        XpsDocument doc = new XpsDocument(filename,
FileAccess.Read);
        viewer.Document = doc.GetFixedDocumentSequence();
        //showWindow is a new window file in ShowWindow.xaml
        Window ShowWindow = new Window();
        ShowWindow.Width = this.Width/2;
        ShowWindow.Height = this.Height/2;
        ShowWindow.Content = viewer;
        ShowWindow.Show();
      }
      catch (Exception ex)
      {
        MessageBox.Show(ex.Message, "Error!");
      }
    }
  }
}
```

SkirtBuildViewModel.cs

```
using DiplomWork.Models;
using Prism.Commands;
using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Media;
namespace DiplomWork.ViewModels
{
  interface IBuiltSkirtViewModel
  {
    Action Open3DModel { get; set; }
  }
  public class BuiltSkirtViewModel: INotifyPropertyChanged,
IBuiltSkirtViewModel
  {
```

```
    private ModelBuiltSkirt _model;
    private DelegateCommand _open3DModelCommand;
    public DelegateCommand open3DModelCommand =>
_open3DModelCommand ?? (_open3DModelCommand = new
DelegateCommand(Open3DModelComm));
    public void Open3DModelComm()
    {
      Open3DModel?.Invoke();
    }
    public Action Open3DModel { get; set; }
    public BuiltSkirtViewModel()
    {
      _model = new ModelBuiltSkirt();
    }
    public PathGeometry CoordinatGeom
    {
      get
      {
        return _model.PathGeom;
      }
    }
    public double DpiOneCm
    {
      set
      {
        _model.OneCm = value * 37.7952755905512;
      }
    }
    public MeasurementsModel Measurements
    {
      set
      {
        _model.Measurements = value;
        OnPropertyChanged("Measurements");
      }
```

```
      get
      {
        return _model.Measurements;
      }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged([CallerMemberName] string prop
= "")
    {
      if (PropertyChanged != null)
        PropertyChanged(this, new
PropertyChangedEventArgs(prop));
    }
  }
}
```

SkirtBuiltModel.cs

```
using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows;
using System.Windows.Media;
namespace DiplomWork.Models
{
  public class ModelBuiltSkirt : INotifyPropertyChanged
  {
    Point W = new Point(oneCm * 5, oneCm * 5);
    Point W1;
    Point W2;
    Point W21;
    Point W22;
    Point W3;
    Point W31;
    Point W32;
    Point W33;
```

```
Point W4;
Point W41;
Point W42;
Point W43;
//additional
Point W5;
Point W51;
Point W52;
Point W53;
Point W6;
Point W61;
Point W62;
Point W63;
Point H;
Point H1;
Point H2;
Point B;
Point B1;
Point B2;
double WidthCon;
double increaseHipp = 0 * oneCm;
double increaseWaist = 0 * oneCm;
double incrSpine;
double incrSpine2;
double incrSide;
double incrFront;
double incrFront2;
int highHip = 0;
private PathGeometry pathGeom;
private MeasurementsModel measurements;
private static double oneCm = 37.7952755905512;
//private static double oneSm = 10;
public ModelBuiltSkirt()
{
  pathGeom = new PathGeometry();
```

```
}
public PathGeometry PathGeom
{
  get
  {
    StartBuild();
    return pathGeom;
  }
}
public double OneCm
{
  set
  {
    oneCm = value;
    OnPropertyChanged("OneCm");
  }
  get
  {
    return OneCm;
  }
}
public double IncreaseWaist
{
  set
  {
    if (value <= 2)
    {
      increaseWaist = value * oneCm;
      OnPropertyChanged("IncreaseWaist");
    }
  }
}
public double IncreaseHipp
{
  set
```

```
      {
        if (value <= 3)
        {
          increaseHipp = value * oneCm;
          OnPropertyChanged("IncreaseHipp");
        }
      }
    }
    public int HighHip
    {
      set
      {
        highHip = value;
        OnPropertyChanged("HighHip");
      }
    }
    public MeasurementsModel Measurements
    {
      set
      {
        measurements = value;
        OnPropertyChanged("Measurements");
      }
      get
      {
        return measurements;
      }
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public void OnPropertyChanged([CallerMemberName] string prop
= "")
    {
      if (PropertyChanged != null)
        PropertyChanged(this, new
PropertyChangedEventArgs(prop));
```

```
    }
    // РЕАЛИЗАЦИЯ АЛГОРИТМА ПОСТРОЕНИЯ ЮБКИ
    private void StartBuild()
    {
      WidthCon = Convert.ToDouble(measurements.Hh) * oneCm +
increaseHipp;
      // 2 stage
      H = new Point(W.X, W.Y + (18 + highHip) * oneCm);
      pathGeom.Figures.Add(LineSegm(W,H));
      // 3 stage
      //ternar operation
      B = new Point(H.X, W.Y + Convert.ToDouble(measurements.Dl)
* oneCm);
      pathGeom.Figures.Add(LineSegm(H,B));
      //4 stage
      //Bottom
      B1 = new Point(B.X + WidthCon, B.Y);
      pathGeom.Figures.Add(LineSegm(B,B1));
      //Hipp
      H1 = new Point(H.X + WidthCon, H.Y);
      pathGeom.Figures.Add(LineSegm(H,H1));
      //Waist
      W1 = new Point(W.X + WidthCon, W.Y);
      pathGeom.Figures.Add(LineSegm(W,W1));
      //from T1 to B1;
      pathGeom.Figures.Add(LineSegm(W1,B1));
      //5 stage
      if(Convert.ToDouble(measurements.Hh)<=53&&
Convert.ToDouble(measurements.Hw)<=42)
      {
        W2 = new Point((W1.X - W.X) / 2 + W.X, W.Y);
      }
      else
      {
        W2 = new Point((W1.X - W.X) / 2 - oneCm + W.X, W.Y);
```

```
      }
      H2 = new Point(W2.X, H.Y);
      B2= new Point(W2.X, B.Y);
      pathGeom.Figures.Add(LineSegm(W2,B2));
      //6 stage
      TuckCalculation();
      #region 7-9 stage Draw Tucks
      //вытачка по боку
      W21 = new Point(W2.X - incrSide / 2, W2.Y - oneCm);
      W22 = new Point(W2.X + incrSide / 2, W2.Y - oneCm);
      pathGeom.Figures.Add(LineSegm(W21, new Point(W21.X, W21.Y
+ oneCm)));
      pathGeom.Figures.Add(LineSegm(W22, new Point(W22.X,
W22.Y+oneCm)));
      pathGeom.Figures.Add(LineSegm(W21, H2));
      pathGeom.Figures.Add(LineSegm(W22,H2));
      if (incrSpine2 == 0)
      {
        // вытачка по спинке
        W3 = new Point((W21.X - W.X) / 2 + W.X, W.Y);
        W31 = new Point(W3.X - incrSpine / 2, W3.Y);
        W32 = new Point(W3.X + incrSpine / 2, W3.Y);
        W33 = new Point(W3.X, W.Y + (13+highHip)*oneCm);
        pathGeom.Figures.Add(LineSegm(W31,W33));
        pathGeom.Figures.Add(LineSegm(W3,W33));
        pathGeom.Figures.Add(LineSegm(W32,W33));
        // вытачка по полочке
        W4 = new Point((W1.X - W22.X) / 2 + W22.X, W.Y);
        W41 = new Point(W4.X - incrFront / 2, W4.Y);
        W42 = new Point(W4.X + incrFront / 2, W4.Y);
        int addittionalH = 0;
        if (incrFront > 2 * oneCm)
          addittionalH = 2;
        else if (incrFront > 1 * oneCm)
          addittionalH = 1;
```

```
        W43 = new Point(W4.X, W.Y + (8 + addittionalH) * oneCm);
        pathGeom.Figures.Add(LineSegm(W41,W43));
        pathGeom.Figures.Add(LineSegm(W4,W43));
        pathGeom.Figures.Add(LineSegm(W42,W43));
      }
      else
      {
        ///От т.Т вправо от 9 до 10 см получаем точку Т3 от Т3
вправо и влево больший раствор вытачки деленных на 2 , вниз от
13 до 15 см.
        ///Затем отрезок Т3 Т21 делим на 2 получаем точку Т5 ,от
нее вправо и в лево меньший раствор вытачки деленный на два вниз
перепендикуляр
        ///на 1 до 1.5 см меньше чем предыдущая
        // вытачка по спинке
        W3 = new Point(W.X+9*oneCm, W.Y);
        W31 = new Point(W3.X - incrSpine / 2, W3.Y);
        W32 = new Point(W3.X + incrSpine / 2, W3.Y);
        W33 = new Point(W3.X, W.Y + (13 + highHip) * oneCm);
        W5 = new Point((W21.X - W3.X) / 2 + W3.X, W3.Y);
        W51 = new Point(W5.X - incrSpine2 / 2, W3.Y);
        W52 = new Point(W5.X + incrSpine2 / 2, W3.Y);
        W53 = new Point(W5.X, W33.Y - oneCm);
        pathGeom.Figures.Add(LineSegm(W31,W33));
        pathGeom.Figures.Add(LineSegm(W3,W33));
        pathGeom.Figures.Add(LineSegm(W32,W33));
        pathGeom.Figures.Add(LineSegm(W51,W53));
        pathGeom.Figures.Add(LineSegm(W5,W53));
        pathGeom.Figures.Add(LineSegm(W52,W53));
        W4 = new Point(W1.X - 9 * oneCm, W.Y);
        W41 = new Point(W4.X - incrFront / 2, W4.Y);
        W42 = new Point(W4.X + incrFront / 2, W4.Y);
        int addittionalH = 0;
        if (incrFront > 2 * oneCm)
          addittionalH = 2;
```

```
        else if (incrFront > 1 * oneCm)
          addittionalH = 1;
        W43 = new Point(W4.X, W.Y + (8 + addittionalH) * oneCm);
        W6 = new Point((W4.X - W22.X) / 2 + W22.X, W4.Y);
        W61 = new Point(W6.X - incrFront2 / 2, W6.Y);
        W62 = new Point(W6.X + incrFront2 / 2, W6.Y);
        W63 = new Point(W6.X, W43.Y - oneCm);
        pathGeom.Figures.Add(LineSegm(W41,W43));
        pathGeom.Figures.Add(LineSegm(W4,W43));
        pathGeom.Figures.Add(LineSegm(W42,W43));
        pathGeom.Figures.Add(LineSegm(W61, W63));
        pathGeom.Figures.Add(LineSegm(W6, W63));
        pathGeom.Figures.Add(LineSegm(W62,W63));
      }
      #endregion
      //10 stage
      double W21H2 = Math.Sqrt(Math.Pow(H2.X - W21.X, 2) +
Math.Pow(H2.Y - W21.Y, 2));
      double W21X1 = W21H2 / 2 + oneCm / 2; //1
      double k = W21X1 / W21H2;
      Point X0L = new Point((W21.X + H2.X) / 2, (W21.Y + H2.Y) /
2);
      Point x1 = new Point(W21.X + (H2.X - W21.X) * k, W21.Y +
(H2.Y - W21.Y) * k);
      x1 = new Point(X0L.X + (x1.Y - X0L.Y), X0L.Y - (x1.X -
X0L.X));
      Point X0R = new Point((W22.X + H2.X) / 2, (W22.Y + H2.Y) /
2);
      Point x2 = new Point(W22.X + (H2.X - W22.X) * k, W22.Y +
(H2.Y - W22.Y) * k);
      x2 = new Point(X0R.X - (x2.Y - X0R.Y), X0R.Y + (x2.X -
X0R.X));

      pathGeom.Figures.Add(LineSegm(x1, X0L));
      pathGeom.Figures.Add(LineSegm(x2, X0R));
```

```
      pathGeom.Figures.Add(BezierSegm(W21, x1, new Point(H2.X,
(H2.Y - x1.Y) * 0.4 + x1.Y), H2));
      pathGeom.Figures.Add(BezierSegm(W22, x2, new Point(H2.X,
(H2.Y - x2.Y) * 0.4 + x2.Y), H2));
      //11 stage
      pathGeom.Figures.Add(QuadraticBezierSegm(W32, new
Point((W21.X - W32.X) * 3 / 4 + W32.X, W32.Y), W21));
      pathGeom.Figures.Add(QuadraticBezierSegm(W22, new
Point((W42.X - W22.X) / 4 + W22.X, W42.Y), W42));
    }
    // РАССЧЕТ ВЫТОЧЕК
    private void TuckCalculation()
    {
      incrSpine2 = 0;
      incrFront2 = 0;
      double sum = Math.Round( WidthCon + increaseHipp -
(Convert.ToDouble(measurements.Hw)*oneCm + increaseWaist));
      incrSpine = Math.Round(sum * 0.3, 1);
      incrSide = Math.Round(sum * 0.5, 1);
      incrFront = Math.Round(sum * 0.2, 1);
      double maxSp = 4.0 *oneCm;
      double maxS = 7.0 *oneCm;
      double maxF = 3.0 * oneCm;
      if (sum > 14 * oneCm)
      {
        maxS = 8.0 * oneCm;
      }
      if (sum<=15 * oneCm)
      {
        if (incrSpine > maxSp)
        {
          if (incrFront + (incrSpine - maxSp) <= maxF)
            incrFront += incrSpine - maxSp;
          else if (incrSide + (incrSpine - maxSp) <= maxS)
```

```
            incrSide += incrSpine - maxSp;
          incrSpine = maxSp;
        }
        else if (incrSide > maxS)
        {
          if (incrFront + (incrSide - maxS) <= maxF)
            incrFront += incrSide - maxS;
          else if (incrSpine + (incrSide - maxS) <= maxSp)
            incrSpine += incrSide - maxS;
          incrSide = maxS;
        }
        else if (incrFront > maxF)
        {
          if (incrSide + (incrFront - maxF) <= maxS)
            incrSide += incrFront - maxF;
          else if (incrSpine + (incrFront - maxF) <= maxSp)
            incrSpine += incrFront - maxF;
          incrFront = maxF;
        }
      }
      else
      {
        if (incrSide > maxS)
        {
          incrFront += incrSide - maxS;
          incrSide = maxS;
        }
        incrSpine2 = Math.Round(incrSpine/oneCm * 0.38,1)*oneCm;
        incrSpine = Math.Round(incrSpine / oneCm * 0.62,1) *
oneCm;
        incrFront2 = Math.Round(incrFront / oneCm * 0.38,1) *
oneCm;
        incrFront = Math.Round(incrFront / oneCm * 0.62,1) *
oneCm;
      }
```

```
}
    private PathFigure LineSegm(Point A, Point B)
    {
      LineSegment etLS = new LineSegment();
      PathFigure etPF = new PathFigure();
      etPF.StartPoint = A;
      etLS.Point = B;
      etPF.Segments.Add(etLS);
      return etPF;
    }
    private PathFigure BezierSegm(Point start, Point p1, Point
p2, Point p3)
    {
      BezierSegment t1 = new BezierSegment();
      t1.Point1 = p1;
      t1.Point2 = p2;
      t1.Point3 = p3;
      PathFigure etPF = new PathFigure();
      etPF.StartPoint = start;
      etPF.Segments.Add(t1);
      return etPF;
    }
    private PathFigure QuadraticBezierSegm(Point start, Point
p1, Point p2)
    {
      QuadraticBezierSegment t1 = new QuadraticBezierSegment();
      t1.Point1 = p1;
      t1.Point2 = p2;
      PathFigure etPF = new PathFigure();
      etPF.StartPoint = start;
      etPF.Segments.Add(t1);
      return etPF;
    }
  }
}
```

## 4 Enums

```
namespace DiplomWork
{
  public enum EnumTypeClothes
  {
    Skirt,
    Throusers,
    Dress,
    None
  }
}
namespace DiplomWork.Enum
{
  public enum EnumTypeConstructionMethods
  {
    CRIGI, //Central Research Institute of Garment Industry
ЦНИИШП
  }
}
```

# ДОДАТОК В

## Діаграма реалізації MVVM у застосунку

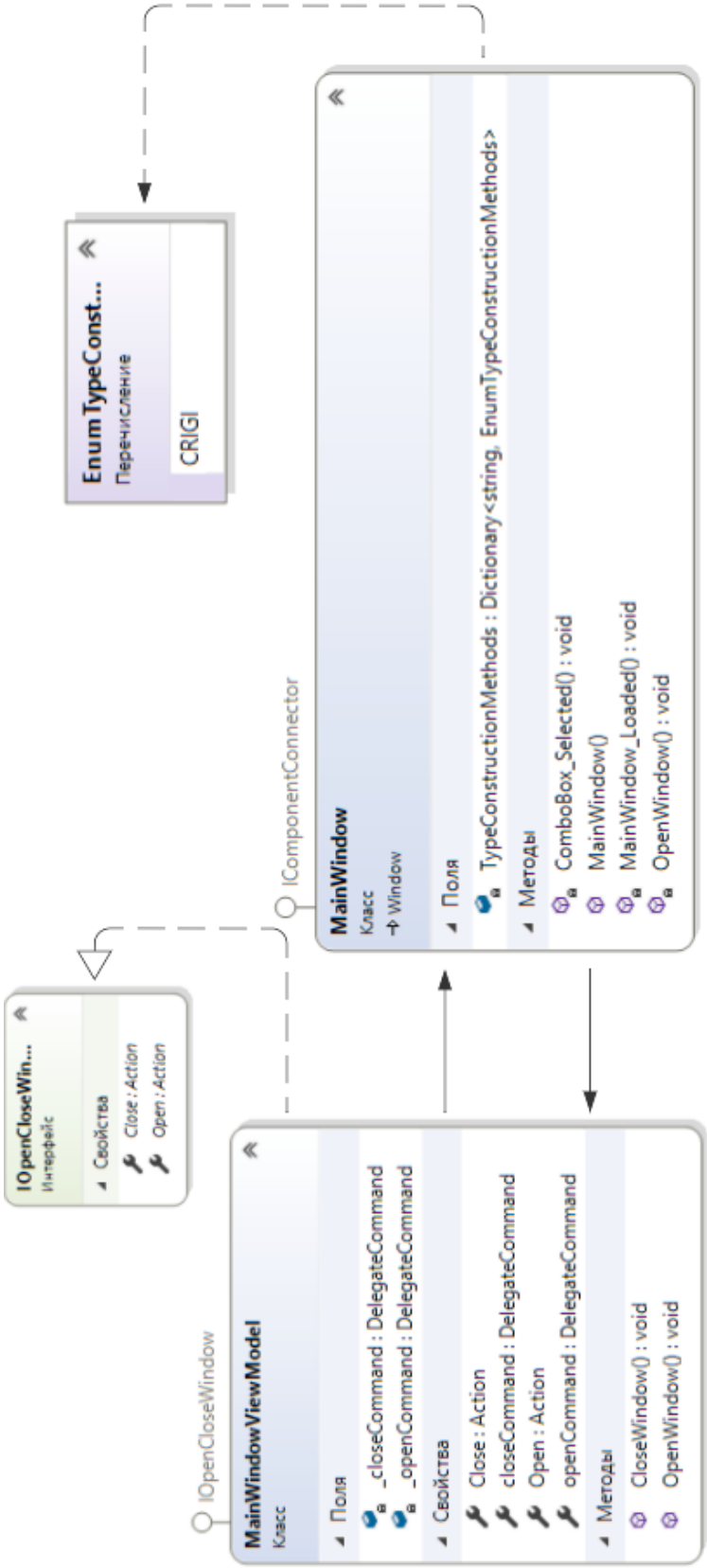


Рисунок В.1 Діаграма реалізації MVVM для MainWindow

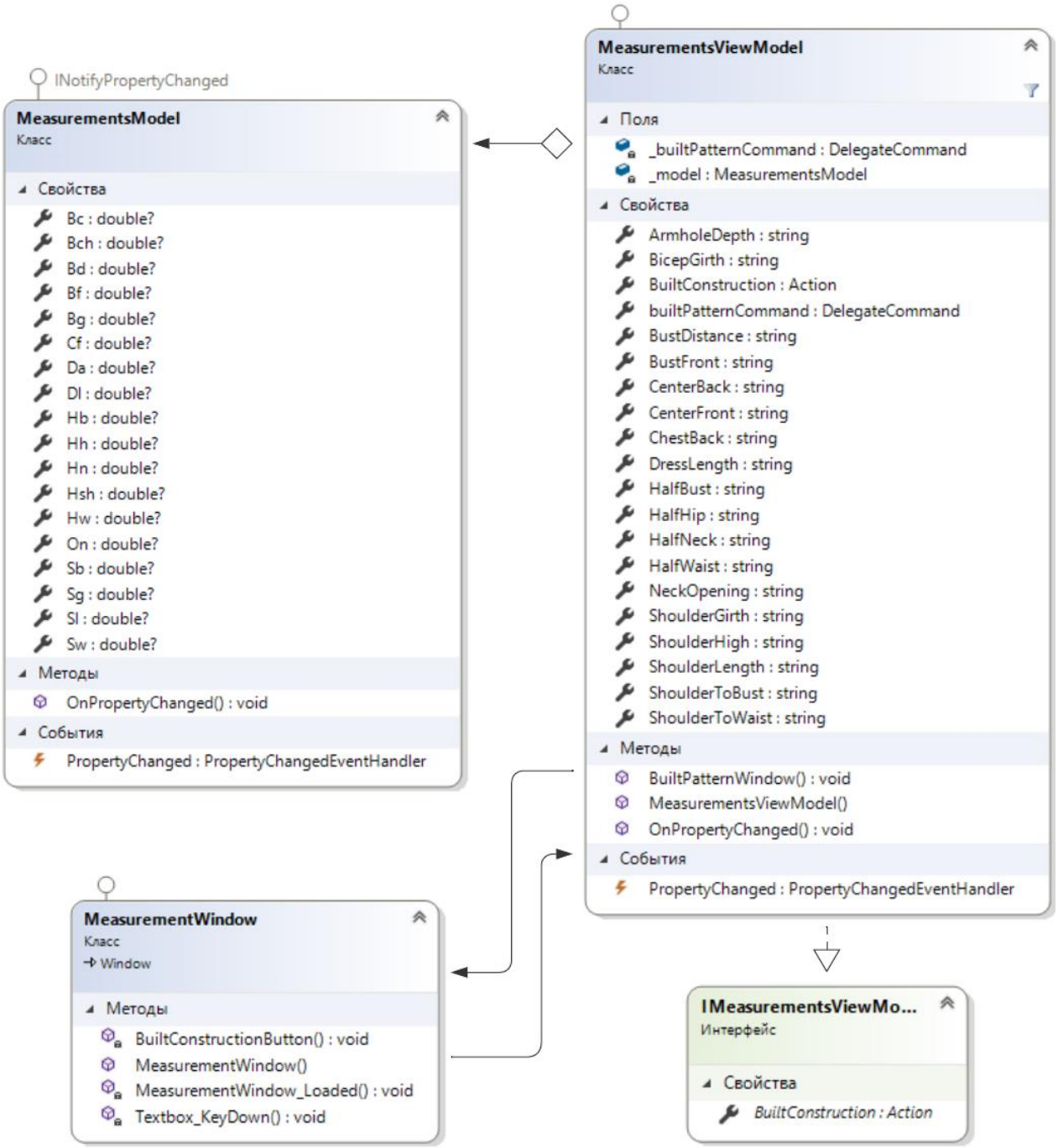


Рисунок В.2 Діаграма реалізації MVVM для MeasurementWindow

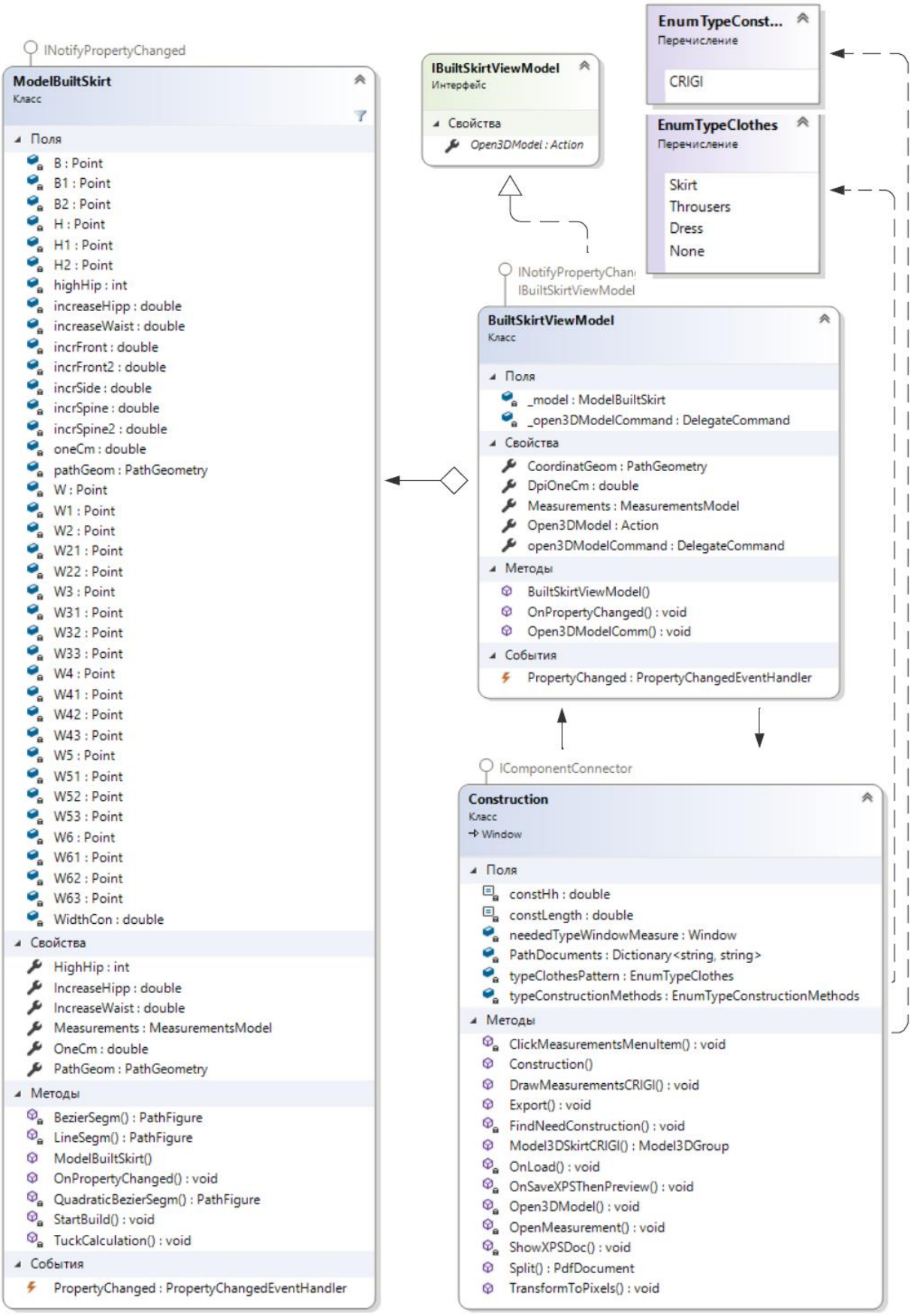


Рисунок В.3 Діаграма реалізації MVVM для Construction

# ДОДАТОК Г

## Довідка про впровадження

Подписано цифровой
подписью: Olga Roznovets
Дата: 2022.06.21 11:47:14
+03'00'

# ДОВІДКА

## про впровадження

Інформаційна система для побудови 2D та 3D лекал одягу, розроблена студенткою Одеського національного університету імені І.І. Мечникова Жмакіною Анастасією Семенівною під час виконання дипломної роботи бакалавра на кафедрі математичного забезпечення комп'ютерних систем, прийнята до використання в клубі крою та шиття «Tavifa».

ФОП Штодько Елена Михайлівна _____
(підпис)