# Загальний опис клієнтської частини застосунку

Компонент розподіленої системи, що відіграє роль клієнтського застосунку – «IronMacbeth.Client» (надалі, у цьому розділі, «клієнт»). Інформація щодо функціоналу та шляхів застосування вказана у розділі «».



Мал № . Логотип Windows Presentation Foundation

Клієнт розроблений за технологією WPF (Windows Presentation Foundation) (мал № .). WPF – платформа для створення застосунків з графічним користувацьким інтерфейсом. WPF був розроблений на заміну технології Windows Forms. У порівнянні з Windows Forms, WPF дозволяє розробку сучасних графічних інтерфейсів з використанням адаптивної розмітки; має значну перевагу у тому, що дозволяє розробляти правильно структуровані програми, у яких відображення логічно відділене від моделі, бізнес-логіки і контенту (завдяки використанню прив’язок (англ. Binding) та контексту даних (англ. DataContext)); апаратне прискорення, підтримка мультисенсорного вводу; включає в себе підтримку декларативної мови розмітки XAML (англ. eXtensible Application Markup Language).

XAML дозволяє створювати витончені та комплексні графічні інтерфейси не вдаючись до імперативного програмування. XAML відкриває можливість описати складні графічні інтерфейси лаконічним і зрозумілим шляхом. XAML розроблений з урахуванням здатності розширюватись, таким чином додавання нових графічних елементів не завдає додаткових складнощів. Кожен елемент у мові розмітки відповідає класу у .Net Framework, що дозволяє безшовно взаємодіяти з елементами графічного інтерфейсу у C# класах (за необхідності).

Також у клієнтському застосунку було використано шаблон проектування MVVM (англ. Model-View-ViewModel). Даний підхід дозволяє цілковито відділити відображення (View) від логіки взаємодії (ViewModel), яка вносить зміни у стан об’єктів бізнес логіки (Model).Таким чином досягаються наступні переваги застосунку:

* Відображення (View) може розроблятись абсолютно відокремлено від логіки взаємодії (ViewModel), а логіка взаємодії та модель (Model) абсолютно незалежно від відображення.
* Тестованість. Адже логіка взаємодії не має жодних залежностей від відображення, така логіка легко піддається юніт-тестуванню, що підвищує кінцеву якість застосунку.
* Перевикористовуваність. Логіка взаємодії та відображення не має жорсткої зав’язки один на одного, а отже виникає можливість перевикористовувати логіку взаємодії або відображення з декількома різними відображеннями\різною логікою взаємодії.