**Тема: «Многопользовательское приложение»**

**Практическая часть**

Продолжаем дополнение задания 14 «Технологии создания графического пользовательского интерфейса»

Каждая задача должна содержать три ключевых аспекта:

1. **Применение MVVM** – создание модели, ViewModel и связывание их с View.
2. **Разделение логики и представления** – вынос бизнес-логики из UI в отдельные классы.
3. **Асинхронное программирование** – выполнение фоновых задач (загрузка данных, работа с сетью, базой данных).

**1. АРМ работника склада**

✅ **MVVM:**

* Создать ProductModel, WarehouseModel для хранения товаров и склада.
* Разработать WarehouseViewModel, содержащую коллекцию товаров (ObservableCollection<ProductModel>).
* Реализовать команды (ICommand) для добавления/удаления товаров.

✅ **Разделение логики и представления:**

* В WarehouseService реализовать методы загрузки данных.
* В ViewModel использовать WarehouseService, а не обращаться к базе данных напрямую.

✅ **Асинхронное программирование:**

* Реализовать асинхронную загрузку списка товаров с имитацией задержки.
* Кнопка "Загрузить товары" должна отключаться при загрузке.

**2. CRM для небольшого бизнеса**

✅ **MVVM:**

* Создать ClientModel, OrderModel.
* Реализовать CRMViewModel с коллекцией клиентов и заказов.
* Создать команды для добавления нового клиента, редактирования и удаления.

✅ **Разделение логики и представления:**

* Вынести бизнес-логику работы с клиентами в ClientService.
* В View оставить только DataBinding и визуальное оформление.

✅ **Асинхронное программирование:**

* Имплементировать загрузку списка клиентов через Task.Run().
* Добавить индикатор загрузки.

**3. Электронный дневник студента**

✅ **MVVM:**

* Создать StudentModel, GradeModel.
* Реализовать StudentViewModel, содержащую список студентов и их оценки.
* Добавить команды для выставления оценок.

✅ **Разделение логики и представления:**

* Вынести логику работы с оценками в GradeService.
* В View оставить ListBox для отображения студентов.

✅ **Асинхронное программирование:**

* Сделать загрузку списка студентов асинхронной.
* При загрузке показывать индикатор.

**4. Управление задачами (Task Manager)**

✅ **MVVM:**

* Создать TaskModel, TaskCategoryModel.
* Реализовать TaskManagerViewModel с коллекцией задач и их категорий.
* Добавить команды для создания и удаления задач.

✅ **Разделение логики и представления:**

* Вынести управление задачами в TaskService.
* В XAML оставить только ListView с DataTemplate.

✅ **Асинхронное программирование:**

* Реализовать отложенную загрузку задач (await Task.Delay(2000)).
* Добавить анимацию загрузки.

**5. Система бронирования отеля**

✅ **MVVM:**

* Создать RoomModel, BookingModel.
* Реализовать HotelViewModel с коллекцией доступных номеров.
* Команды: "Забронировать", "Отменить бронирование".

✅ **Разделение логики и представления:**

* Вынести логику бронирования в BookingService.
* В XAML оставить ListView номеров и Button.

✅ **Асинхронное программирование:**

* При бронировании номера ожидать подтверждения (await Task.Delay(3000)).
* Показать индикатор загрузки.

**6. Управление заказами в интернет-магазине**

✅ **MVVM:**

* Создать ProductModel, OrderModel.
* Реализовать ShopViewModel с коллекцией заказов.
* Добавить команды для оформления заказа.

✅ **Разделение логики и представления:**

* Вынести обработку заказов в OrderService.
* В XAML использовать DataGrid с шаблонами.

✅ **Асинхронное программирование:**

* Имитация обработки заказа (await Task.Delay(3000)).
* Показывать индикатор выполнения.

**7. Автоматизация учета сотрудников**

✅ **MVVM:**

* Создать EmployeeModel, DepartmentModel.
* Реализовать EmployeeViewModel с коллекцией сотрудников.
* Добавить команды для редактирования данных сотрудников.

✅ **Разделение логики и представления:**

* Вынести бизнес-логику учета сотрудников в EmployeeService.
* В XAML использовать ComboBox для выбора отдела.

✅ **Асинхронное программирование:**

* Имплементировать загрузку сотрудников через Task.Run().
* Добавить ProgressBar при загрузке данных.

**8. Приложение для учета финансов**

✅ **MVVM:**

* Создать TransactionModel, CategoryModel.
* Реализовать FinanceViewModel с коллекцией транзакций.
* Добавить команды для фильтрации данных.

✅ **Разделение логики и представления:**

* Вынести обработку транзакций в FinanceService.
* В XAML использовать Chart для отображения расходов.

✅ **Асинхронное программирование:**

* Загрузка данных из API (await Task.Delay(3000)).
* Обновление UI после получения данных.

**9. Бронирование билетов в кинотеатр**

✅ **MVVM:**

* Создать SeatModel, MovieModel.
* Реализовать CinemaViewModel с коллекцией мест.
* Добавить команды для бронирования мест.

✅ **Разделение логики и представления:**

* Вынести управление бронированием в CinemaService.
* В XAML использовать UniformGrid для отображения мест.

✅ **Асинхронное программирование:**

* Имитация подтверждения бронирования (await Task.Delay(2000)).
* Показать всплывающее уведомление после успешного бронирования.

**10. Система контроля версий документов**

✅ **MVVM:**

* Создать DocumentModel, VersionModel.
* Реализовать VersionControlViewModel с историей версий документов.
* Добавить команды для сохранения новой версии.

✅ **Разделение логики и представления:**

* Вынести управление версиями в VersionService.
* В XAML использовать TreeView для отображения структуры версий.

✅ **Асинхронное программирование:**

* Асинхронная загрузка списка версий.
* Показать анимацию при загрузке.

**11. Программа для учета медицинских записей**

✅ **MVVM:**

* Создать PatientModel, MedicalRecordModel.
* Реализовать MedicalRecordsViewModel с коллекцией пациентов и их историей болезни.
* Добавить команды для добавления новой медицинской записи.

✅ **Разделение логики и представления:**

* Вынести управление медицинскими записями в MedicalRecordService.
* В XAML использовать DataGrid для списка пациентов.

✅ **Асинхронное программирование:**

* Имплементировать загрузку истории болезни (await Task.Delay(2000)).
* Показать индикатор загрузки при открытии карточки пациента.

**12. Управление учебным процессом (электронный журнал преподавателя)**

✅ **MVVM:**

* Создать StudentModel, CourseModel, GradeModel.
* Реализовать JournalViewModel с коллекцией студентов, курсов и оценок.
* Добавить команды для редактирования оценок.

✅ **Разделение логики и представления:**

* Вынести бизнес-логику работы с курсами и оценками в JournalService.
* В XAML использовать ListView с DataTemplate.

✅ **Асинхронное программирование:**

* Имитация загрузки списка студентов (await Task.Delay(3000)).
* Добавить прогресс-бар во время загрузки.

**13. Приложение для аренды автомобилей**

✅ **MVVM:**

* Создать CarModel, RentalModel.
* Реализовать CarRentalViewModel с коллекцией доступных автомобилей.
* Добавить команды для аренды и возврата автомобиля.

✅ **Разделение логики и представления:**

* Вынести управление бронированием в RentalService.
* В XAML использовать ComboBox для выбора автомобиля.

✅ **Асинхронное программирование:**

* Реализовать имитацию обработки заявки (await Task.Delay(4000)).
* Показать статус "Обрабатывается" во время загрузки.

**14. Приложение для организации мероприятий**

✅ **MVVM:**

* Создать EventModel, ParticipantModel.
* Реализовать EventViewModel с коллекцией мероприятий и участников.
* Добавить команды для регистрации на мероприятие.

✅ **Разделение логики и представления:**

* Вынести управление мероприятиями в EventService.
* В XAML использовать CalendarView для отображения мероприятий.

✅ **Асинхронное программирование:**

* Имитация отправки приглашения (await Task.Delay(3000)).
* Отобразить уведомление после успешной регистрации.

**15. Электронная библиотека**

✅ **MVVM:**

* Создать BookModel, AuthorModel.
* Реализовать LibraryViewModel с коллекцией книг и авторов.
* Добавить команды для поиска и бронирования книг.

✅ **Разделение логики и представления:**

* Вынести логику управления библиотекой в LibraryService.
* В XAML использовать ListBox для отображения списка книг.

✅ **Асинхронное программирование:**

* Реализовать асинхронный поиск книг (await Task.Delay(1500)).
* Добавить индикатор загрузки при получении результатов.