# Произдственная практика по специальности 09.02.07 Информационные системы и программирование

Тема: «Разработка информационно-справочной системы «Персональные компьютеры»»

Выполнил студент группы ИС-32 Суворов А.А.

**Цель проекта** - разработка информационно-справочной системы о Персональных компьютерах.

**Объект исследования** - информационно-справочные системы.

**Предмет исследования -** особенности реализации информационно-справочной системы, а также средства реализации программного продукта.

## Для достижения поставленной цели в данном проекте были решены следующие задачи:

- •определены темы, которые будут реализованы
- ■спроектированы структура и функциональность приложений;
- изучены и выбраны средства разработки информационносправочных систем;
- •разработана графическая составляющая приложени1;
- ■реализована структура и функциональность Программного продукта;

#### Выбор Инструментальных средств







#### Мобильная версия



Экран загрузки

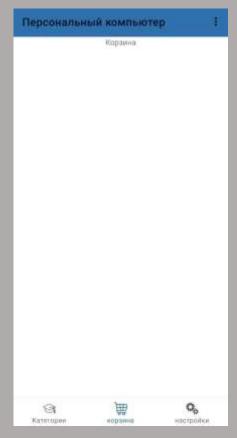


Главное меню

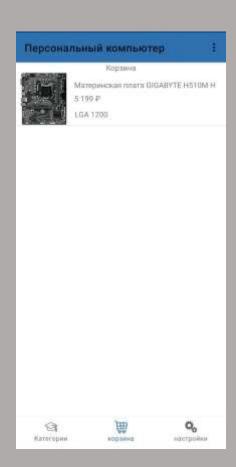


Настройки

### Экран с корзиной



Пустая корзина



Корзина после покупки

#### Экран с выбранной лекцией

#### Персональный компьютер

платы зависит срок работы компьютера, а от её конфигурации — возможность улучшать его в

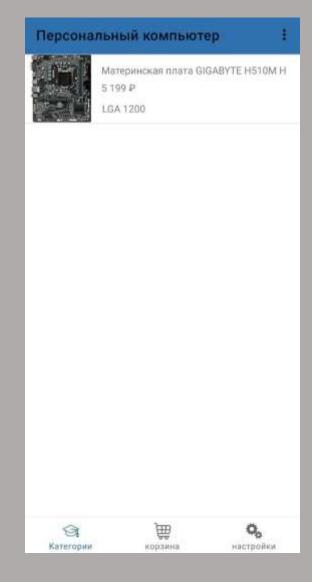
будущем. Материнскую плату выбирают с учётом сокета процессора, чипсета, количества и вида слотов для остальных элементов системы Форм-фактор материнской платы — стандарт, определяющий размеры материнской платы для компьютера, места её крепления к шасси; расположение на ней интерфейсов шин, портов ввода-вывода, разъёма процессора, слотов для оперативной памяти, а также тип разъёма для подключения блока питания. Форм-фактор (как и любые другие стандарты) носит рекомендательный характер. Спецификация форм-фактора определяет обязательные и опциональные компоненты. Однако подавляющее большинство производителей предпочитают соблюдать спецификацию, поскольку ценой соответствия существующим стандартам является совместимость материнской платы и стандартизированного оборудования (периферии, карт расширения) других производителей (что имеет ключевое значение для снижения стоимости владения, англ. TCO). Устаревшими являются форматы: Baby-AT; полноразмерная плата АТ; LPX; BTX, MicroBTX и PicoBTX. Современные и массово применяемые форматы: ATX; microATX; Mini-ITX. Внедряемые форматы: Nano-ITX; Pico-ITX; FlexATX; NLX; WTX, CEB. Существуют материнские платы, не соответствующие никаким из существующих форм-факторов. Это принципиальное решение производителя, обусловленное желанием создать на рынке несовместимый с существующими продуктами «бренд» (Apple, Commodore, Silicon Graphics, Hewlett-Packard, Compaq чаще других игнорировали

ПЕРЕЙТИ К ПОКУПКЕ

стандарты) и эксклюзивно производить к нему периферийные устройства и аксессуары.



### Экран покупки



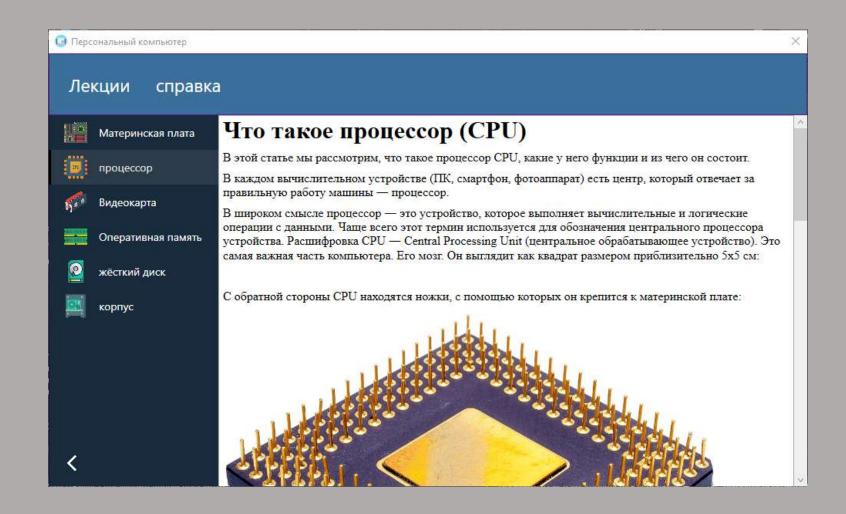
#### Оптимизированная передача данных

```
class ViewModel1 : ViewModel() {
      val name : MutableLiveData<String> by lazy {
          MutableLiveData<String>()
      val cena: MutableLiveData<String> by lazy {
          MutableLiveData<String>()
      val spec: MutableLiveData<String> by lazy {
          MutableLiveData<String>()
      val img: MutableLiveData<Int> by lazy {
          MutableLiveData<Int>()
```

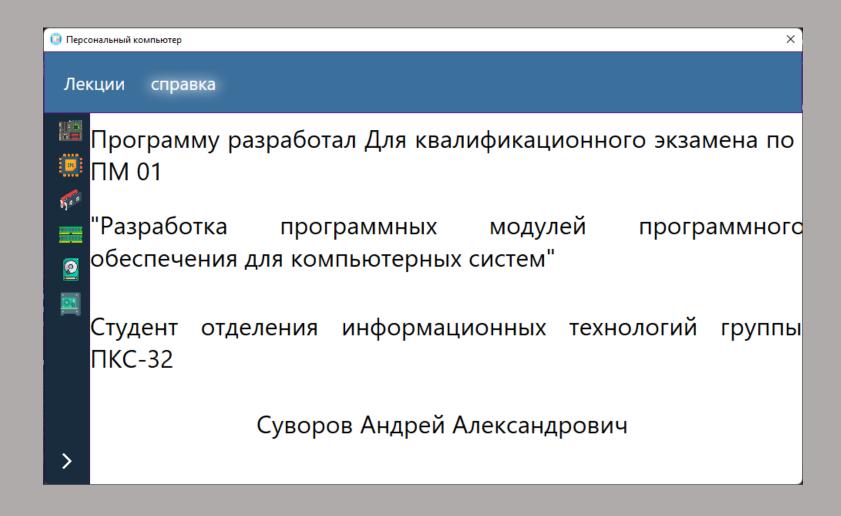
#### Передача списка

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
  lateinit var adapter: CustomAdapter
  val viewModels : ViewModel1 by activityViewModels()
  val name : String = viewModels.name.value.toString()
  val cena : String = viewModels.cena.value.toString()
  val spec : String = viewModels.spec.value.toString()
  val img : Int = viewModels.img.value!!.toInt()
  val listView: ListView = view.findViewById<View>(R.id.spisok_pokupok) as ListView
  val dataModels: ArrayList<dataModel> = ArrayList()
      dataModels.add(
          dataModel(
               img,
               name,
               cena,
               spec
  adapter = CustomAdapter(dataModels, requireActivity().application)
  listView.adapter = adapter
   super.onViewCreated(view, savedInstanceState)
```

#### Desktop-версия



#### Окно справки



#### Взаимодействие с интерфейсом

```
Label Content="{TemplateBinding Content}"
     Background=["Transparent" FontSize="23"
     VerticalAlignment="Center">
  <Label.Style>
      <Style TargetType="Label">
          <Setter Property="Foreground" Value=""White"/>
          <Style.Triggers>
              <DataTrigger Binding="{Binding Path=IsMouseOver, ElementName=Btn}"</pre>
                            Value="true">
                  <Setter Property="Foreground" Value=="White"/>
              </DataTrigger>
              <DataTrigger Binding="{Binding RelativeSource={RelativeSource}</pre>
                  Mode=FindAncestor,
                  AncestorType={x:Type Button}},
                 Path = IsPressed}" Value="true">
                  <Setter Property="Foreground" Value=="White"/>
              </DataTrigger>
              <DataTrigger Binding="{Binding RelativeSource={RelativeSource}</pre>
                  Mode=FindAncestor,
                  AncestorType={x:Type Button}},
                 Path = IsFocused} " Value="true">
                  <Setter Property="Foreground" Value=="White"/>
               </DataTrigger>
          </Style.Triggers>
      </Style>
  </Label.Style>
Label>
```

#### Регистрация действий пользователя

```
mamespace personal_computer.ViewModel
    Ссылок: 1
    internal abstract class ViewModelBase : INotifyPropertyChanged
        public event PropertyChangedEventHandler PropertyChanged;
        Ссылокс 1
        protected virtual void onPropertyChanged([CallerMemberName] string PropertyName = null)
            PropertyChanged? Invoke(this, new PropertyChangedEventArgs(PropertyName)):
        Ссылок: 1
        protected virtual bool Set<T>(ref T field, T value, [CallerMemberName] string PropertyName = null)
            if (Equals(field, value)) return false;
            field = value;
            onPropertyChanged(PropertyName);
            return true;
```