ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

ассистент		И.Д. Свеженин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

Проектирование современного программного обеспечения с использованием Jetpack Compose

по курсу: Кроссплатформенное программирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1. Цель работы:

Выполнить проектирование и разработку мобильного приложения под ОС Android на языке программирования высокого уровня Kotlin.

2. Задание:

1. «Интерактивная карта».

Разработайте приложение с интерактивной картой, используя Google Maps API или Яндекс.Карты API. Добавьте возможность маркировки мест на карте, а также отображение информации о выбранном месте. Карта разрабатывается для города в соответствии городом по варианту.

2. Оптимизируйте перебор массива используя встроенные средства языка программирования Kotlin и протестируйте на массивах от 100000 элементов. Выведите в Activity исходный, итоговый и промежуточные массивы.

Вариант 16

Город - Волгоград.

3. Краткое описание хода разработки, алгоритма работы программы и назначение используемых технологий

Ход разработки:

1. **Анализ задания**:

Требуется создать Android-приложение на Kotlin с двумя функциями:

- Интерактивная карта для Волгограда с возможностью добавлять маркеры и просматривать информацию о них.
- Оптимизированная обработка массива из 100 000 элементов с выводом результатов в интерфейс.

2. **Выбор технологий**:
   - Решено использовать Jetpack Compose — современный декларативный фреймворк для построения UI без XML.
   - Для карты выбран Google Maps API через библиотеку Google Maps Compose, так как она обеспечивает стабильную интеграцию с Compose и не требует сложной настройки (в отличие от Яндекс.MapKit в учебных условиях).
3. **Реализация**:
   a. Созданы три экрана: Карта, Избранное (сохранённые маркеры), Тест массива.
   b. Добавлена навигация между экранами с помощью FloatingActionButton.
   c. Реализовано сохранение маркеров в SharedPreferences.
   d. Для части 2 — сгенерирован массив из 100 000 элементов, применена оптимизация через asSequence(), результаты выведены в UI.
4. **Тестирование**:
   a. Проверена работа карты: добавление, удаление маркеров, геокодирование адреса.
   b. Проверка, что обработка 100 000 элементов не вызывает остановки приложения и корректно отображается.

Используемые технологии:

Kotlin

Основной язык разработки — безопасный, лаконичный, поддерживает функциональные конструкции (Sequence, лямбды).

Jetpack Compose

Современный UI-фреймворк от Google. Позволяет создавать интерфейс декларативно, без XML, с полной поддержкой реактивности.

Google Maps Compose

Официальная библиотека для встраивания Google Maps в Compose-приложения. Упрощает работу с картой, маркерами и событиями.

SharedPreferences

Простое хранилище для сохранения списка маркеров между запусками приложения (в формате строк с разделителями).

Geocoder

Стандартный API Android для преобразования координат в человекочитаемый адрес.

Sequence (Kotlin)

Обеспечивает ленивую обработку больших коллекций — ключевая часть оптимизации перебора массива (избегает создания промежуточных списков).

Описание структуры работы частей приложения:

Приложение состоит из двух логически независимых, но интегрированных частей, каждая из которых реализует одно из двух заданий методических указаний.

1. Часть «Интерактивная карта»

Назначение:

Предоставить пользователю возможность просматривать карту города Волгограда, добавлять интересующие его места в виде маркеров и получать информацию о них.

Структура и работа:

- При запуске приложения открывается экран с Google Maps, центрированной на координатах Волгограда (48.7080° с.ш., 44.5156° в.д.).
- При нажатии на любую точку карты появляется диалоговое окно для ввода названия места.
- После подтверждения:
    - Приложение автоматически определяет адрес по координатам с помощью Geocoder.
    - Создаётся объект FavoritePlace (с уникальным ID, названием, координатами и адресом).
    - Маркер отображается на карте и сохраняется в локальном хранилище (SharedPreferences).
- При нажатии на существующий маркер открывается диалог с возможностью удалить запись.
- Все сохранённые места можно просмотреть на втором экране — «Избранное», где представлен список с названием, координатами и адресом.

Используемые технологии: Jetpack Compose, Google Maps API, Google Maps Compose, Geocoder, SharedPreferences.

2. Часть «Оптимизация перебора массива»

Назначение:

Продемонстрировать эффективную обработку больших объёмов данных с использованием встроенных средств языка Kotlin.

Структура и работа:

- При переходе на третий экран — «Тест массива» автоматически запускается обработка данных.
- Создаётся исходный массив из 100 000 элементов типа FavoritePlace с синтетическими данными (название, случайные координаты в районе Волгограда, адрес).

- Применяется оптимизированный перебор с использованием:
  - .asSequence() — для ленивой (отложенной) обработки, что снижает потребление памяти,
  - .filter { ... } — оставляет только элементы, чьи координаты попадают в границы Волгограда (48.5–49.0° ш., 44.0–45.0° д.),
  - .sortedBy { it.id } — сортирует результат по уникальному идентификатору,
  - .toList() — финализирует обработку и возвращает итоговый список.
- В интерфейсе отображаются:
  - Исходный массив (первые 5 элементов),
  - Промежуточный массив (после фильтрации),
  - Итоговый массив (после сортировки).

Соответствие заданию:

- Использованы встроенные средства Kotlin (Sequence, функциональные операторы),
- Объём данных — ровно 100 000 элементов,
- Результаты выведены в Activity (в UI, а не только в лог).

Используемые технологии: Kotlin (Sequence API, лямбда-выражения), Jetpack Compose (для отображения результатов).

🔄 Интеграция частей

Обе части объединены в единое приложение с удобной навигацией:

- Карта → Избранное → Тест массива → обратно к карте.
- Навигация реализована через FloatingActionButton, что обеспечивает интуитивное управление.
- Несмотря на логическую независимость, обе части используют общую модель данных (FavoritePlace), что демонстрирует целостность архитектуры.

4. Скриншоты, иллюстрирующие результаты работы программы:

В качестве демонстрации работы программы приведены несколько скриншотов.

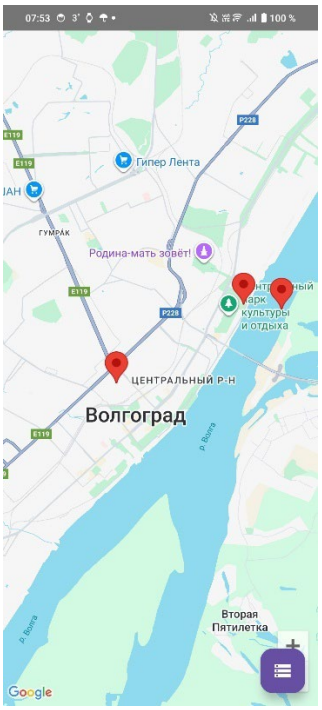

Рисунок 1 – Стартовый экран

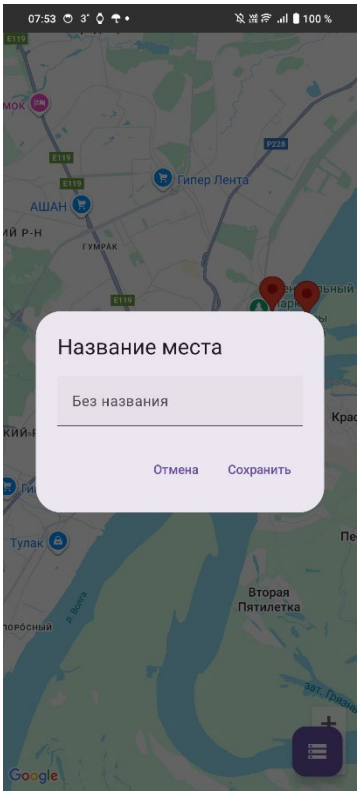

Рисунок 2.1 – Добавление маркера

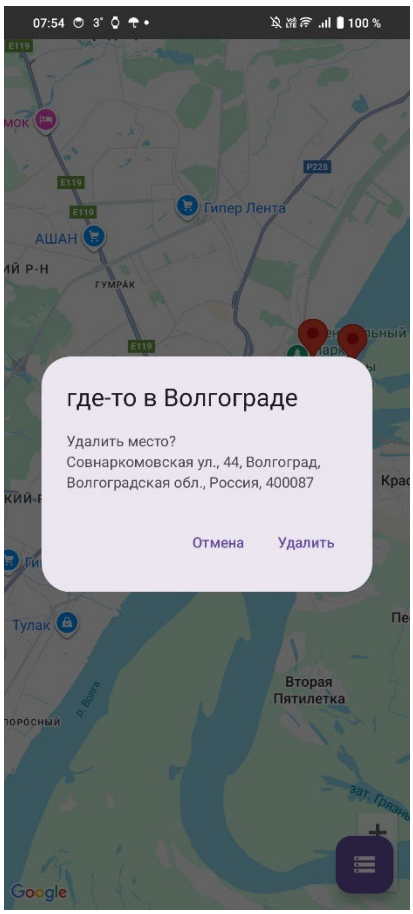

Рисунок 2.2 – Удаление маркера

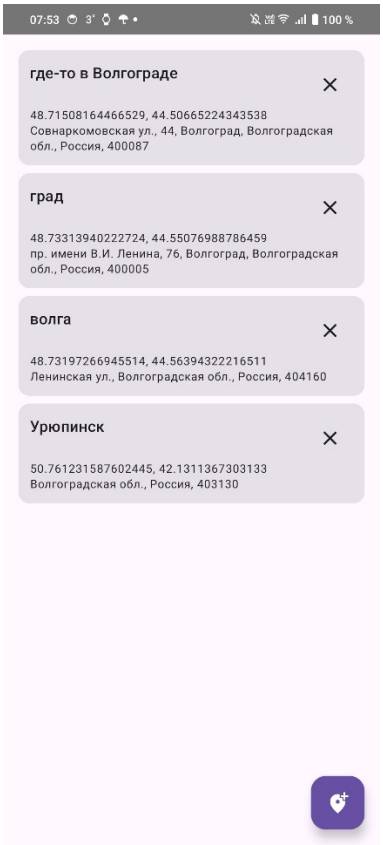

Рисунок 2.3 – Список маркеров

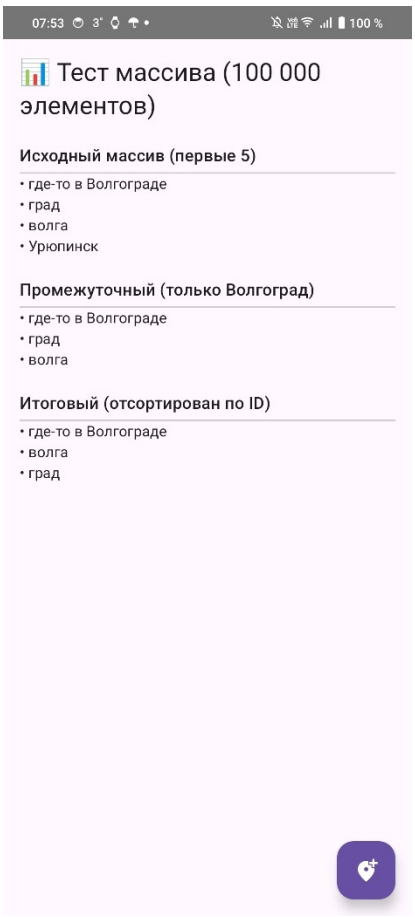

Рисунок 3.1 – Обработка массива маркеров

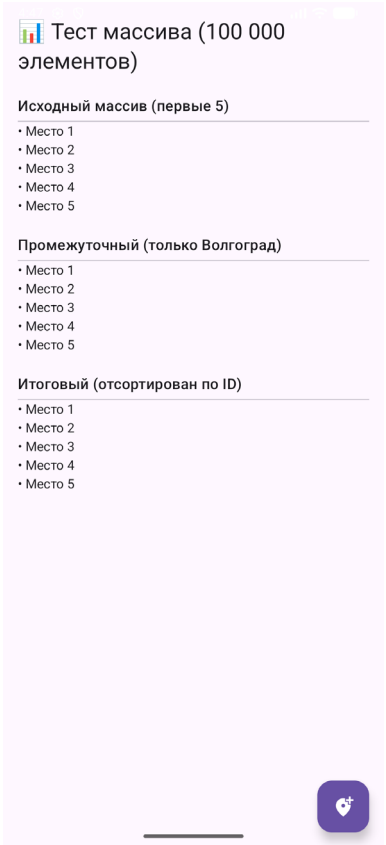

Рисунок 3.2 – Тест обработки массива 100 000 элементов

ВЫВОД

В ходе выполнения работы было разработано мобильное Android-приложение на языке Kotlin, темой которого является попугаи. Приложение реализует функционал хранения, добавления, просмотра, обновления и удаления данных о попугаях с использованием локальной базы данных Room, а также получение и парсинг JSON из интернета в отдельном потоке. Реализована навигация по фрагментам с помощью BottomNavigationView, а также отображение данных в списке через RecyclerView с возможностью обработки долгого нажатия и вызова диалогового окна с действиями.

Проект успешно демонстрирует современный подход к разработке Android-приложений, включающий работу с асинхронными операциями, безопасное хранение данных, корректную обработку ошибок и удобный пользовательский интерфейс. Все поставленные задачи были выполнены: создана база данных, реализован CRUD-функционал, интегрирована работа с JSON, настроена навигация и отображение данных. Приложение стабильно работает, корректно обрабатывает дубликаты данных и обеспечивает удобное взаимодействие с пользователем.

Разработка позволила получить практические навыки работы с такими технологиями, как Room, RecyclerView, Coroutines, Gson, Navigation, и закрепить знания по архитектуре и жизненному циклу Android-приложений.

ПРИЛОЖЕНИЕ А

Листинг программы

MainActivity.kt:

```
package com.example.a71r

import android.os.Bundle
import android.location.Geocoder
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.AddLocationAlt
import androidx.compose.material.icons.filled.Close
import androidx.compose.material.icons.filled.Storage
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import com.google.android.gms.maps.model.LatLng
import com.google.maps.android.compose.*
import android.content.Context
import com.google.android.gms.maps.model.CameraPosition
import kotlinx.coroutines.*

// === Data ===
data class FavoritePlace(
    val id: Long,
    val name: String,
    val lat: Double,
    val lng: Double,
    val address: String
)

// === MainActivity ===
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MaterialTheme {
                Surface(modifier = Modifier.fillMaxSize()) {
                    MainApp()
                }
            }
        }
    }
}

@Composable
fun MainApp() {
    var screen by remember { mutableStateOf("map") }
    when (screen) {
        "map" -> MapScreen { screen = "favorites" }
        "favorites" -> FavoritesScreen { screen = "array" }
        "array" -> ArrayTestScreen { screen = "map" }
    }
```

```
}

// === Карта ===
@Composable
fun MapScreen(onFavoriteClick: () -> Unit) {
    val context = LocalContext.current
    val prefs = context.getSharedPreferences("markers", Context.MODE_PRIVATE)
    var markers by remember { mutableStateOf(loadMarkers(prefs)) }
    var dialogState by remember {
mutableStateOf<DialogState>(DialogState.Closed) }

    val cameraPositionState = rememberCameraPositionState {
        position = CameraPosition.fromLatLngZoom(LatLng(48.7080, 44.5156),
12f) // Волгоград
    }

    Box(modifier = Modifier.fillMaxSize()) {
        GoogleMap(
            modifier = Modifier.fillMaxSize(),
            cameraPositionState = cameraPositionState,
            onMapClick = { latLng ->
                dialogState = DialogState.NameInput(latLng, "")
            }
        ) {
            markers.forEach { m ->
                Marker(
                    state = MarkerState(LatLng(m.lat, m.lng)),
                    title = m.name,
                    snippet = m.address,
                    onClick = {
                        dialogState = DialogState.DeleteConfirm(m)
                        true
                    }
                )
            }
        }

        FloatingActionButton(
            onClick = onFavoriteClick,
            containerColor = MaterialTheme.colorScheme.primary,
            modifier = Modifier
                .align(Alignment.BottomEnd)
                .padding(16.dp)
        ) {
            Icon(Icons.Default.Storage, "Избранное")
        }

        when (val state = dialogState) {
            is DialogState.NameInput -> {
                NameInputDialog(
                    latLng = state.latLng,
                    initialName = state.name,
                    onConfirm = { name ->
                        val address = getAddress(context,
state.latLng.latitude, state.latLng.longitude)
                        val id = System.currentTimeMillis()
                        val place = FavoritePlace(id, name,
state.latLng.latitude, state.latLng.longitude, address)
                        saveMarker(place, prefs)
                        markers = markers + place
                        dialogState = DialogState.Closed
                    },
                    onDismiss = { dialogState = DialogState.Closed }
```

```
                )
            }
            is DialogState.DeleteConfirm -> {
                DeleteDialog(
                    place = state.place,
                    onConfirm = {
                        removeMarker(state.place.id, prefs)
                        markers = markers.filter { it.id != state.place.id }
                        dialogState = DialogState.Closed
                    },
                    onDismiss = { dialogState = DialogState.Closed }
                )
            }
            DialogState.Closed -> {}
        }
    }
}

// === Избранное ===
@Composable
fun FavoritesScreen(onArrayClick: () -> Unit) {
    val context = LocalContext.current
    val prefs = context.getSharedPreferences("markers", Context.MODE_PRIVATE)
    var markers by remember { mutableStateOf(loadMarkers(prefs)) }

    Box(modifier = Modifier.fillMaxSize()) {
        LazyColumn(
            modifier = Modifier.fillMaxSize(),
            contentPadding = PaddingValues(16.dp),
            verticalArrangement = Arrangement.spacedBy(8.dp)
        ) {
            items(markers, key = { it.id }) { place ->
                FavoriteItem(place) {
                    removeMarker(place.id, prefs)
                    markers = loadMarkers(prefs)
                }
            }
        }

        FloatingActionButton(
            onClick = onArrayClick,
            containerColor = MaterialTheme.colorScheme.primary,
            modifier = Modifier
                .align(Alignment.BottomEnd)
                .padding(16.dp)
        ) {
            Icon(Icons.Default.AddLocationAlt, "Тест")
        }
    }
}

@Composable
fun FavoriteItem(place: FavoritePlace, onDelete: () -> Unit) {
    Card(modifier = Modifier.fillMaxWidth()) {
        Column(modifier = Modifier.padding(12.dp)) {
            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.SpaceBetween
            ) {
                Text(place.name, style =
MaterialTheme.typography.titleMedium)
                IconButton(onClick = onDelete) {
                    Icon(Icons.Default.Close, "Удалить")
```

```
                }
            }
            Text("${place.lat}, ${place.lng}\n${place.address}", style =
MaterialTheme.typography.bodySmall)
        }
    }
}

// === Экран теста массива (ШАГ 3) ===
@Composable
fun ArrayTestScreen(onBack: () -> Unit) {
    val context = LocalContext.current
    val prefs = context.getSharedPreferences("markers", Context.MODE_PRIVATE)
    var original by remember { mutableStateOf<List<String>>(emptyList()) }
    var intermediate by remember { mutableStateOf<List<String>>(emptyList())
}
    var final by remember { mutableStateOf<List<String>>(emptyList()) }

    LaunchedEffect(Unit) {
        // Получаем данные: либо из маркеров, либо генерируем 100 000
        val places = if (prefs.contains("places")) {
            loadMarkers(prefs)
        } else {
            (1..100000).map { i ->
                FavoritePlace(
                    id = i.toLong(),
                    name = "Место $i",
                    lat = 48.7 + (kotlin.random.Random.nextDouble() - 0.5) *
0.4,
                    lng = 44.5 + (kotlin.random.Random.nextDouble() - 0.5) *
0.4,
                    address = "Адрес $i"
                )
            }
        }

        // Исходный — первые 5
        original = places.take(5).map { it.name }

        // Промежуточный — после фильтрации (только Волгоград)
        val filtered = places.asSequence()
            .filter { p -> p.lat in 48.5..49.0 && p.lng in 44.0..45.0 }
            .toList()
        intermediate = filtered.take(5).map { it.name }

        // Итоговый — после сортировки
        val sorted = filtered.asSequence()
            .sortedBy { it.id }
            .toList()
        final = sorted.take(5).map { it.name }
    }

    Box(modifier = Modifier.fillMaxSize()) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp)
                .verticalScroll(rememberScrollState()),
            verticalArrangement = Arrangement.spacedBy(16.dp)
        ) {
            Text("📊 Тест массива (100 000 элементов)", style =
MaterialTheme.typography.headlineSmall)
```

```
            Section("Исходный массив (первые 5)", original)
            Section("Промежуточный (только Волгоград)", intermediate)
            Section("Итоговый (отсортирован по ID)", final)
        }

        FloatingActionButton(
            onClick = onBack,
            containerColor = MaterialTheme.colorScheme.primary,
            modifier = Modifier
                .align(Alignment.BottomEnd)
                .padding(16.dp)
        ) {
            Icon(Icons.Default.AddLocationAlt, "Назад")
        }
    }
}

@Composable
fun Section(title: String, items: List<String>) {
    Column {
        Text(title, style = MaterialTheme.typography.titleMedium, modifier =
Modifier.padding(vertical = 4.dp))
        Divider()
        items.forEach { item ->
            Text("• $item", style = MaterialTheme.typography.bodyMedium)
        }
    }
}

// === Диалоги и утилиты ===
sealed class DialogState {
    object Closed : DialogState()
    data class NameInput(val latLng: LatLng, val name: String) :
DialogState()
    data class DeleteConfirm(val place: FavoritePlace) : DialogState()
}

@Composable
fun NameInputDialog(latLng: LatLng, initialName: String, onConfirm: (String)
-> Unit, onDismiss: () -> Unit) {
    var name by remember { mutableStateOf(initialName) }
    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text("Название места") },
        text = {
            TextField(value = name, onValueChange = { name = it },
placeholder = { Text("Без названия") })
        },
        confirmButton = {
            TextButton(onClick = { onConfirm(name.ifBlank { "Без названия" })
}) { Text("Сохранить") }
        },
        dismissButton = { TextButton(onClick = onDismiss) { Text("Отмена") }
}
    )
}

@Composable
fun DeleteDialog(place: FavoritePlace, onConfirm: () -> Unit, onDismiss: () -
> Unit) {
    AlertDialog(
        onDismissRequest = onDismiss,
        title = { Text(place.name) },
```

```
        text = { Text("Удалить место?\n${place.address}") },
        confirmButton = { TextButton(onClick = onConfirm) { Text("Удалить") }
},
        dismissButton = { TextButton(onClick = onDismiss) { Text("Отмена") }
}
    )
}

fun loadMarkers(prefs: android.content.SharedPreferences):
List<FavoritePlace> {
    return prefs.getStringSet("places", emptySet())?.mapNotNull { s ->
        val p = s.split("|")
        if (p.size == 5) FavoritePlace(p[0].toLong(), p[1], p[2].toDouble(),
p[3].toDouble(), p[4]) else null
    } ?: emptyList()
}

fun saveMarker(place: FavoritePlace, prefs:
android.content.SharedPreferences) {
    val set = prefs.getStringSet("places", mutableSetOf())!!.toMutableSet()

set.add("${place.id}|${place.name}|${place.lat}|${place.lng}|${place.address}
")
    prefs.edit().putStringSet("places", set).apply()
}

fun removeMarker(id: Long, prefs: android.content.SharedPreferences) {
    val set = prefs.getStringSet("places", mutableSetOf())!!.toMutableSet()
    set.removeIf { it.startsWith("$id|") }
    prefs.edit().putStringSet("places", set).apply()
}

fun getAddress(context: Context, lat: Double, lng: Double): String {
    return try {
        Geocoder(context).getFromLocation(lat, lng,
1)?.firstOrNull()?.getAddressLine(0) ?: "Адрес не найден"
    } catch (e: Exception) {
        "Ошибка"
    }
}
```