Таким образом, полученные результаты видовой дифференциации *S.aur, S.epid* и *S.haem* в среде ТСБ с высокой степенью надежности (α=95%) показывают возможность автоматизации микробиологического эксперимента, используя для этих целей активную и реактивную составляющие импеданса.

Литература

1. Авдеева, М.Г. Особенности диагностики сепсиса в практике врача-инфекциониста / М.Г. Авдеева, В.Н. Городин, Л.П. Блажняя [и др.] // Эпидемиология и инфекционные болезни. – 2016. – Т. 21, № 1. – С. 4-13.

2. А. И. Драпеза, В. А. Лобан, Н. В. Плешко, Г. А. Скороход, Е. И. Гудкова Импедансные нформационные технологии для ускоренной оценки жизнеспособности микроорганизмов//Вестник БГУ. − Сер.1.− 2015. −№3. − С.24-28.

3. Петри, А. Наглядная медицинская статистика/ А. Петри, К. Сэбин; пер.с англ. под ред. В.П.Леонова. –2-е изд., перераб. и доп. −М.:ГЭОТАР-Медиа,2009. −168с.

Н.С. Середа (ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **В.В. Грищенко**, ст. преподаватель

РАЗРАБОТКА ПРИЛОЖЕНИЯ-АГРЕГАТОРА СЕРВИСОВ ГОРОДА ГОМЕЛЯ ДЛЯ ПЛАТФОРМЫ iOS

Приложение-агрегатор сервисов города Гомеля представляет собой приложение, которое объединяет различного рода сервисы внутри одного приложения. Приложение первоначально включает следующие сервисы:

- сервис радио, для прослушивания локальный радиостанций
- лента локальных новостей;
- различного рода финансовые учреждения в округе (банки, банкоматы);
- расписание транспорта, просмотр маршрутов;

Таким образом приложение нацелено на облегчение навигации внутри города, агрегирование различного рода сервисов и услуг для быстрого доступа к ним. Кроме того, приложение может расширяться, в зависимости от возможностей дополняться новым функционалом.

Перед началом разработки необходимо продумать и разработать архитектуру и базовую структуру проекта. Построение правильной архитектуры программы в дальнейшем существенно облегчает процесс разработки, а также дальнейшей поддержки и расширения существующего функционала, а хорошо продуманная структура существенно облегчает процесс навигации по проекту, что также уменьшает затрачиваемое на разработку.

При разработке API данного приложения были использованы следующие технологии:

- Swift;
- Core data;
- Firebase;
- Jenkins;
- TestFlight;
- Cocoapods.

Начало разработки API представляет собой определение необходимых сторонних библиотек, добавление их посредством Cocoapods - менеджера зависимостей для Swift и Objective-C. Далее следует разработка возможных сценариев, карт переходов между экранами, реализация которых осуществляется при помощи шаблона проектирования Coordinator.

В качестве серверной части, которая отвечает за взаимодействие с внешними системами и получения актуальной информации, используется облачная платформа firebase. Основной сервис firebase – NoSQL база данных реального времени Firebase Realtime Database, которая позволяет хранить и синхронизировать данные между несколькими клиентами. Также немаловажная возможность firebase - Cloud Functions – сервис, позволяющий хранить блоки кода и запускать их при совершении каких-либо событий.

В качестве CI (Continuous integration) используется Jenkins, который отвечает за сборку проекта с определенными настройками (debug, release и т.д). В качестве CD (Continuous delivery) используется TestFlight, отвечающий за размещение сборок, их установки на конечные устройства для последующего тестирования приложения.

В процессе разработки были применены принципы KISS и SOLID. Согласно принципу KISS, большинство систем лучше всего работают, когда остаются простыми. Принципы SOLID представляют собой акроним из пяти основных принципов ООП:

- принцип единой ответственности;
- принцип открытости/закрытости;

- принцип подстановки Барбары Лизков;
- принцип разделения интерфейсов;
- принцип инверсии зависимостей.

После каждого этапа написанный ранее код покрывался юнит-тестами. Это достаточно важный этап в процессе разработки программного обеспечения. В процессе написания кода сложно определить, правильно ли будет работать программа.

Н.П. Скрылев (Белорусско-Российский университет, Могилев)
Науч. рук. **А.И. Якимов,** д-р техн. наук, доцент

АВТОМАТИЗАЦИЯ ПРОТОТИПИРОВАНИЯ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ ПРИ ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ

Процесс разработки комплексной имитационной модели производственно-экономической деятельности промышленного предприятия сопряжён с множеством трудностей [1]. Например, заказчик имитационной модели и ее потенциальные пользователи желают сначала видеть интерфейс, и лишь потом функциональность, в то время как в процессе непосредственно разработки модели сначала создаётся функциональность, а потом для работы с ней создаётся интерфейс. Что из этого следует? Приходится создавать интерфейсы до функциональности для диалога с заказчиком, и в процессе диалога менять их соответственно его желаниям. Но тратить ресурсы на окончательный вариант интерфейса на данном этапе не рационально. Из чего следует вывод: надо создавать не сам интерфейс, а прототипы, или макеты интерфейса.

Макет интерфейса представляет собой графическое изображение того, как будет выглядеть интерфейс в финальной версии продукта. Существуют различные версии макетов интерфейса, но все их объединяет простота создания и изменения; отсутствие осмысленной функциональности. В остальном ограничений на понятие графического макета нет. Это может быть и зарисовка на листе бумаги, и изображение, созданное средствами графического редактора (например, Adobe Photoshop), и даже графический интерфейс окна приложения в редакторе среды программирования (например, Microsoft Visual Studio) без написанной функциональности.