科目名	年度	レポート番号	クラス	学籍番号	名前
API 実習	2021	5	A	20120064	池尻裕輝

ページ数や文字数よりも、読んでわかりやすく書けているかどうかが、点数アップの分かれ目です。

設問(1)

この科目で学んだ内容を第 3 者(他学部の学生や親など)にわかるように説明せよ。

この科目で学んだことは、API についてである。この講義を聞くまで私自身も API についてほぼ初めて聞くものであった。API とはアプリケーションプログラミングインターフェースの略で、ソフトウェアの機能を共有できる仕組みのことを言う。もっと分かりやすく説明すると異なるアプリやソフトウェア同士を繋げる仕組みのことである。この API を利用するメリットとして効率的に開発できる。これは、作りたい機能が既に API で公開されているなら同じプログラムを 1 から作ることがなく開発時間を大幅に短縮できる。また、サービス利用者の利便性が向上する。他社のユーザー情報を使って自社のサービスにログインできる機能を作ることができる。メールアドレスやパスワード入力の必要もないためユーザーにとって面倒な手続きや手間を省くことができる。加えてセキュリティが向上する。Facebook や Twitter、Google の API キーを取得することで、自社のサービスのセキュリティを向上できる。

API には、Web API というものがある。Web API を説明する前に、HTTP/HTTPS について説明する。HTTP とは、Hypertext Transfer Protocol の略で、Web サーバーと Web ブラウザの間で、Web 情報をやりとりするためのプロトコル（通信規則）である。ホームページで情報収集したり、ブログを読んだりする時、この HTTP を使ってサーバーとクライアント（ユーザー）間でやりとりが行われる。HTTP と HTTPS の違いは通信が暗号化されているかいないかの違いであり、HTTPS が暗号化されている通信である。Web API とは、API 提供者と API 利用者とのやりとりを HTTP/HTTPS ベースで実現する API である。Web API は HTTP/HTTPS ベースの API であるため、異なるプログラミング言語で開発されたアプリケーション間を連携させることができる。一方 Web ではない API では API 利用者が用いるプログラミング言語と同じ言語で提供させることが多い。Web API の代表的な実装方法として REST がある。

REST とは、Representational State Transfer の略である。REST とは、システム設計の考え方のことである。ウェブ上のリソース（画像、ファイル、HTML など）へアクセスする際の基本的な考え方が示されてある。例としては「HTTP で、Web のリソースへアクセスができること」「HTTP メソッドを適切に利用すること」「Web のリソースは固有のリソースを持っていること」が挙げられる。

API について習い、次はエンドポイント設計になる。エンドポイントとは API にアクセスするための URI のことである。URI の設計について述べていく。良い URI 設計で重要な原則は、覚えやすく、どんな機能を持つ URI なのかひと目でわかるである。これを意識することで、API を利用する開発者の生産性を向上させ、API の評価が上がる。また、間違ったアクセスが大量に行われ、API を配信するサーバーに負荷がかかってしまう問題も避けることができる。一般的に良いとされる URI の事項として 1 つ目は「ルールが統一されていること」である。利用する単語、URI の構造に一貫性を持たせることで、トラブルや混乱を招くことがなくなる。2 つ目は、「人間が読んで理解できる」である。URI に用いる単語を短くしすぎたり、むやみに省略形を使ったりしないことである。3 つ目は「大文字や小文字を混在させない」である。混在させると分かりづらく間違えやすくなってしまう。標準的に URI には小文字を用いる。4 つ目は「サーバー側のアーキテクチャに依存されない」である。サーバー側のアーキテクチャが反映されている URI は避けるべきである。

では、リソースにアクセスするための URI 設計の注意点について述べようと思う。1 つ目は「複数形の名詞を利用すること」であり、2 つ目は「スペースやエンコードを必要とする文字を使わない」、3 つ目は「2 つ以上の単語をつなげる必要がある場合ハイフンを利用する」

HTTP メソッドと URI の関係性についてである。URI がリソースを表すものだとすると、HTTP メソッドは操作（何をするか）を表すものである。1 つの URI のエンドポイントに異なるメソッドでアクセスすることで、情報を取得するだけでなく情報を変更、削除等さまざまな操作を行うようにすることで、リソースとそれをどう扱うかをきちんと分離して扱うことができる。

クエリパラメタについて説明する。クエリパラメタとは、様々な情報を Web サーバーに伝えるために URL の末尾に付け加える文字列のことである。これを使用する際の注意点は、いくつかある。1 つ目は、「標準的なパラメーターを使用する」である。「＝」の代わりに「：」や「,」を使用したり、「＆」のかわりに「[]」を使用したりする非標準パラメーターを使用すると、検索エンジンにうまく読み取ってもらえないことがある。

認可、OAuth について説明していく。ユーザーのデータを管理する「リソースサーバー」とユーザーのデータを利用したい「クライアントアプリ」があったとする。ユーザーのデータをやりとりする口を API とする。この API を通してクライアントアプリからリソースサーバーにユーザーデータを要求、リソースサーバーからクライアントアプリにユーザーデータを譲渡する。その際、悪意のあるアプリにユーザーのデーターが渡らないようにするのに API を守る仕組みが必要になってくる。API を守る仕組みとして、クライアントアプリにアクセストークンという鍵を持たせておく。アクセストークンにはユーザーのデータを利用する権限が含まれている。ユーザーデータを要求する時に一緒にアクセストークンをリソースサーバーに提示する。リソースサーバーはアクセストークンに含まれている権限を調べ、ユーザーのデータをクライアントアプリに渡す。ここでアクセストークンは初めから持っているものではないため、アクセストークンを発行する係が必要になってくる。アクセストークンを発行する係を認可サーバーという。認可サーバーでアクセストークンを生成し、クライアントアプリに発行する前にユーザーに確認を取り、ユーザーが了承すると無事発行される。この認可サーバーとクライアントアプリの一連の流れを標準化したものを「OAuth2.0」という。

API レスポンスデータの設計や内部構造の考え方について学んだ。レスポンスのフォーマットは 2010 年までは、XML が使われていたが JSON がスタンダードであった。データフォーマットの指定方法は、「クエリパラメーターを使う」「拡張子を使う」「リクエストヘッダでメディアタイプを指定する」の 3 つがある。データの内部構造の考え方は API のアクセス回数がなるべく減るようにすることである。そのために、それぞれの API のユースケースをきちんと考えることが重要になる。レスポンスデータの各項目データ型や項目名を命名する際の注意点は「多くの API で同じ意味に利用されている一般的な単語を用いる」「なるべく少ない単語数で表現する」「変な省略形は極力利用しない」「複数の単語を連結する場合、その連結方法は API 全体を通して統一する」「単数系や複数形に気をつける」である。一般的に良いとされる URI の事項と少し似ている。

次に、ウェブサイトを閲覧するウェブブラウザから送られるリクエストに対して、ウェブサーバーから返信されるレスポンス内容を表す 3 桁の数字である HTTP ステータスコードについて習った。これは、「PC やスマホの画面にこのウェブサイトを表示させたい」というリクエストに対し「現在このサイトは表示できません」などウェブサーバーからのレスポンスを表す数字のことである。この数字は 100～500 番まであり、100 番台はリクエストに対して処理が継続されていることを表している。200 番台はリクエストを正しく受理したことを表している。PC やスマホでウェブサイトを閲覧できている際の多くががこの番号を受信している。300 番台は目的とする情報が別の URI で参照されていることを伝える。400 番台はリクエストが正しく処理できない状態である。ページ自体が存在しないため、サーバーが正しく処理できず、ページが表示できないことになっている。500 番台はサーバーに問題があり、エラーになってしまっている。

キャッシュとプロキシについて説明する。キャッシュとは、表示したウェブページで閲覧したデータなどを一時的に保存し、次回に表示する際にこのデータを使い素早く表示する機能である。プロキシは代理や中継の意味があり、プロキシサーバーはインターネットを接続する際にネットワークの内部と外部を繋ぐ役割を担っている。インターネットに直接接続できないコンピューターの代わりに接続するサーバーである。

API は静的なウェブサイトと違いずっと公開していかなければならない。そのため、設定変更しやすい API の重要性について学んだ。Web API はアプリケーションのインターフェイスとしての役割を持っており、追加、変更、廃止など状況に応じて変化させていく必要がある。しかし、一度公開した Web API の仕様を変更するのは、簡単なことではなく問題が発生する危険性がある。その場合、解決策と

して新しいエンドポイント、あるいは別のパラメータをつけた URI など、何かの新しいアクセス形式で公開すれば良いのである。古い形式でアクセスしてきているクライアントには今までと変わらないデータを送り、新しい形式のアクセスには新しい形式のデータを返せばいいのである。結論、複数のバージョンの API を提供すればいいのである。複数の API を提供するには全く異なる URI で API を公開するのが一番分かりやすく、その際 3 つの方法がある。「URI にバージョンを埋め込む」「バージョンをクエリ文字列に入れる」「メデイアタイプでバージョンを指定する」最もよく利用されるのは URI にバージョンを埋め込む方法である。最もよく利用される方法は URI にバージョンを埋め込む方法である。次に、バージョン変える際の指針について説明していく。API のバージョンは後から変更をしやすくするためのものでもあるが変更をいくら変えてもいいわけではない。バージョンを上げるのはどうしても後方互換性を保ったまま修正を行うことが難しい変更を加えなければならない時かセキュリティや権限などのルールを変更した場合のみである。API の提供を終了する場合はいきなり提供をやめしまうとユーザーはアクセスできない状態になってしまい混乱を招いてしまう。そうしないために終了日をアナウンスするか利用規約にサポート期限を明記しておくことが重要である。

最後に堅牢な API を作るために必要なことを述べていく。API の脅威としては「通信経路上の情報不正入手」「サーバ脆弱性による情報不正入手」「ブラウザアクセスにおける問題」が挙げられる。情報不正入手はパケットスニファリングというパケットの中身を盗み見る行為が行われる。他に中間者攻撃といいクライアントとサーバの中間に入り込んで傍受、盗聴して内容を取得するといった攻撃も存在する。ブラウザアクセスにおける問題としてはセッションハイジャックやクリックジャッキングなどその他にもクロスサイトスクリプティングやクロスサイトリクエストフォージェリ、JSON ハイジャック、HTTP ヘッダインジェクション、SQL インジェクション、OS コマンド、eval インジェクションが挙げられる。API のセキュリティ確保には大きく分けて 2 つの観点を考える必要があり、1 つは API の利用にあたって適切な認証・認可により、データが同意に基づく適切な範囲にのみ流通する仕組みをよのなかで安全と認められた標準的な技術で実現すること。2 つ目に攻撃者が探索のためにリクエストする不正な API もパラメーターの通信を検知しブロックできること。

設問(2)

レポート(4)をもとに、API 連携作成または API を用いたサービス開発結果を書いてください。何かしら動くものが出来ている前提です。

名称

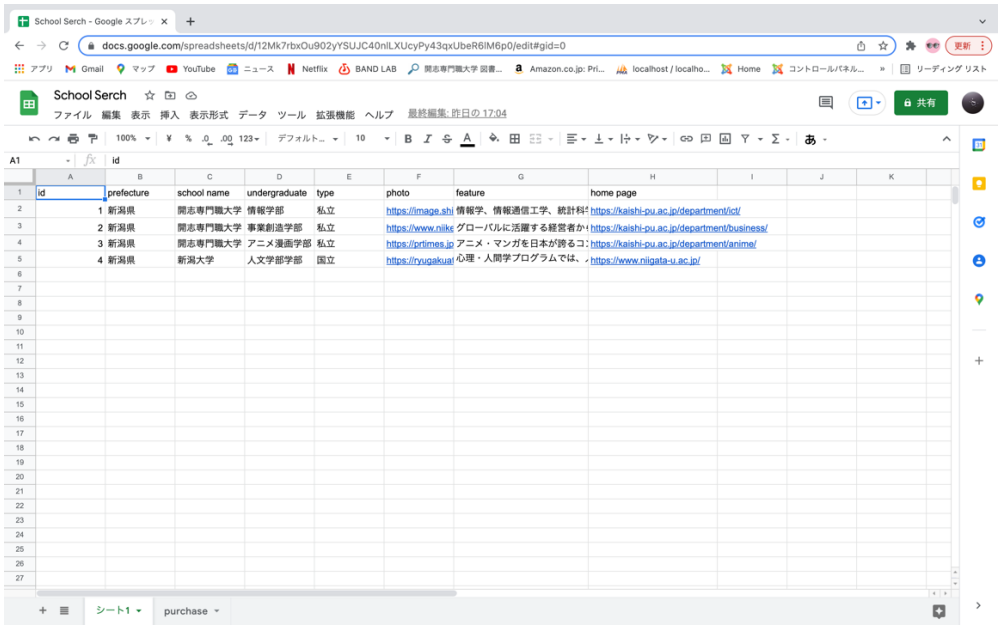
サービス名：school search

概要(作ったものの説明)

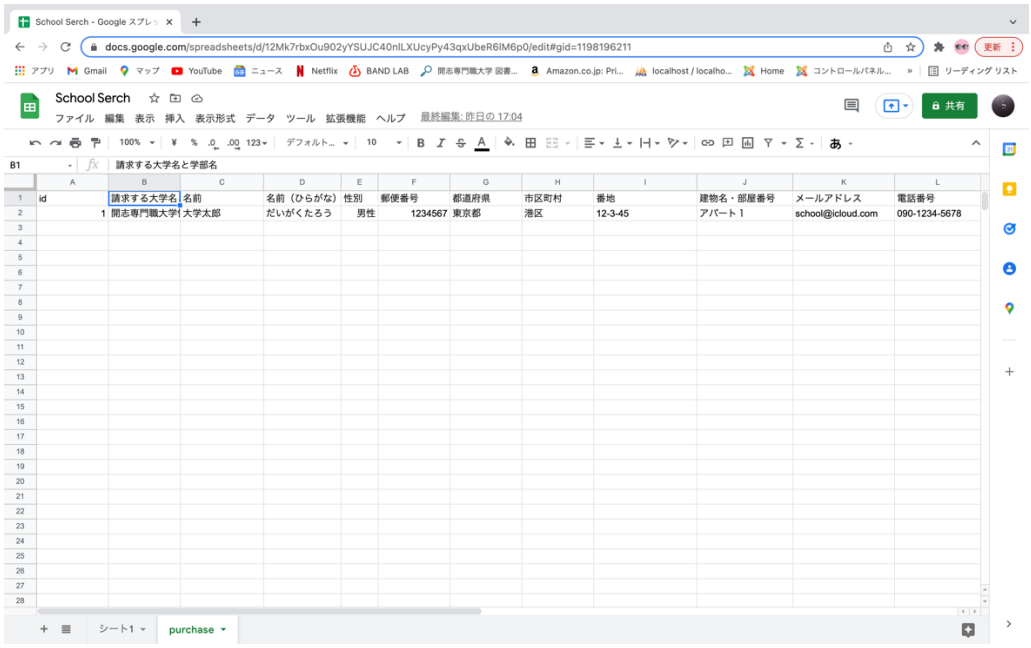
スプレッドシートを使ってスプレッドシートのシート 1 の中に大学情報としてその大学がある県名、大学名、学部学科名、国公私立専門、大学写真、大学の特徴、大学のホームページの情報を入力する。purchase のシートの中に請求する大学名と学部名、ユーザー名、ユーザー名（ひらがな）、性別、郵便番号、ユーザーの都道府県、市区町村、番地、建物名・部屋番号、メールアドレス、電話番号のカラムを作っておく。Sheet. DB で API にした。また、Adaro でアプリを作り、そこに API 連携した。アプリのスタート画面では、会員登録してない場合は、会員登録画面に行き、会員登録している場合はログイン画面に行く。Home 画面は学校一覧になる。学校の写真と大学名と学部・学科名が表示される。学校写真をタップすると学校紹介画面に行く。学校紹介では、スプレッドシートに入力した県名、大学名、学部学科名、国公私立専門、大学写真、大学の特徴、大学のホームページの情報が表示される。ホームページのリンクをタップすると大学のホームページに移動できるようになっている。その下に、資料請求のボタンがあり、そこをタップすると申し込みページに移動し、そこで請求する大学名と学部名、ユーザー名、ユーザー名（ひらがな）、性別、郵便番号、ユーザーの都道府県、市区町村、番地、建物名・部屋番号、メー

ルアドレス、電話番号を入力すると、スプレッドシートに顧客情報が保存され、資料を申し込みできる。申し込みページで情報を入力したら申し込み完了画面に移動し、そこから Home 画面に戻る。

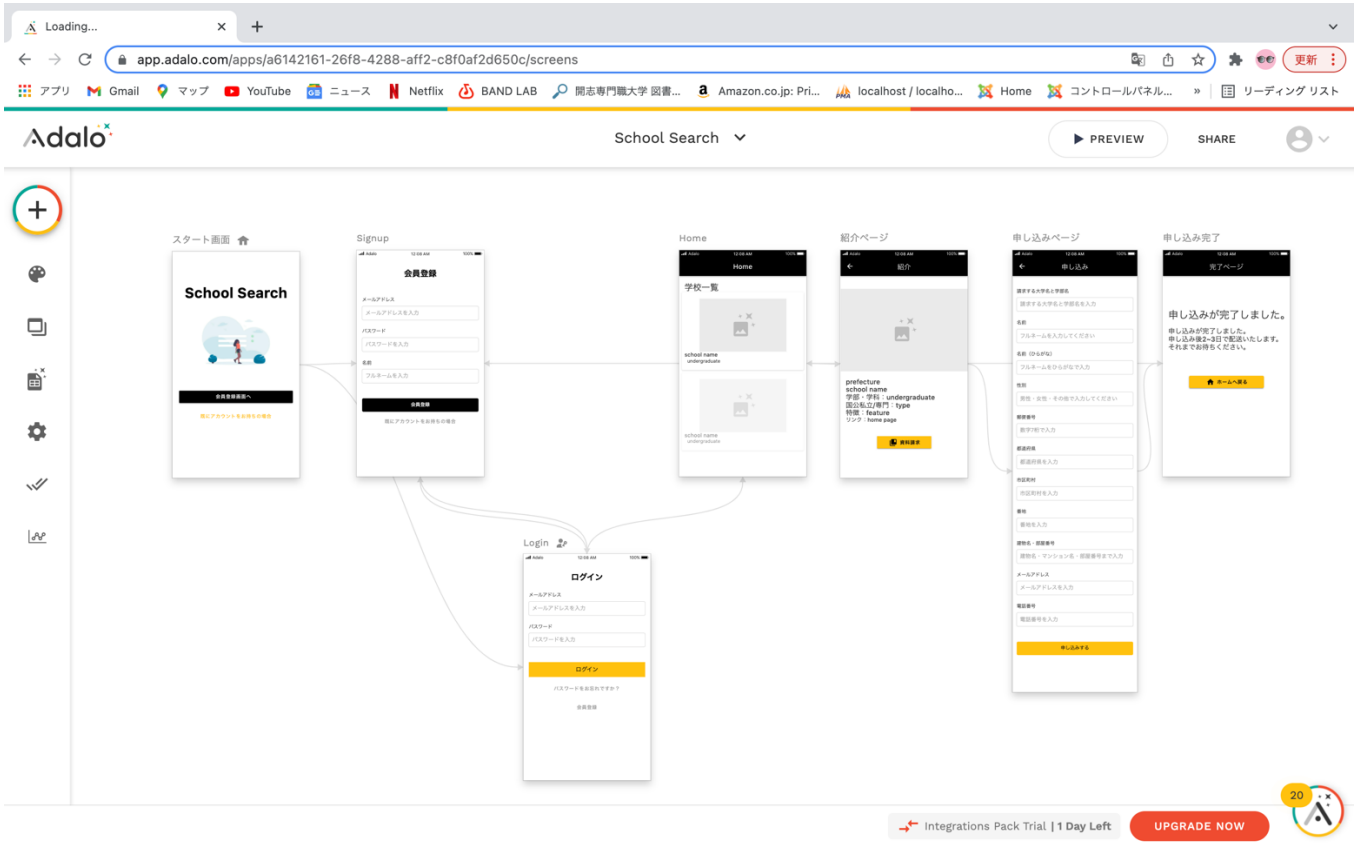
画面ショット(動作がわかるように画面を交えて説明)



スプレッドシートのシート１で大学情報が入力されてある。



purchase シートの中には仮として１行情報を入れておく。



Adalo のアプリ構成一覧

アプリのスタート画面では、会員登録してない場合は、会員登録画面に行き、会員登録している場合はログイン画面に行く。

ログイン画面ではメールアドレスとパスワードを入力する。

パスワードを忘れた場合は「パスワードをお忘れですか？」のところをタップするとパスワードを再設定できるようになる。

学校一覧では学校の写真、学校名、学部学科名が閲覧可能

スプレッドシートに入力しておいて学校写真や学校名、学部名を用いて表示させる。

スプレッドシートのシート１に入力しておいた情報を表示させる。

ここの情報は全部スプレッドシートから持ってきている。あとは Adalo の方で「学部学科：」などの項目を付け足すことで見やすくした。

少し下にスクロールするとリンクと資料請求のボタンがある。

リンクはその大学のホームページに移動できるようになっている。

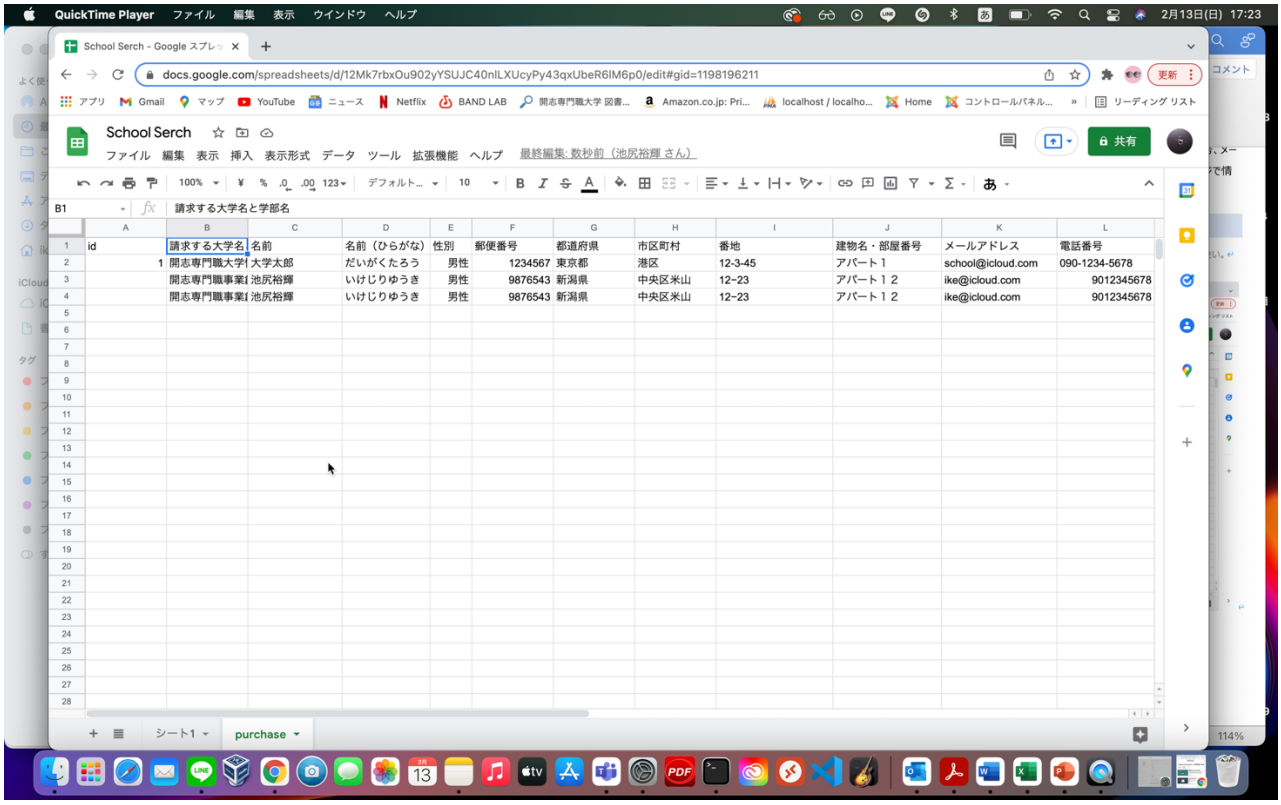
資料請求のボタンをタップすると下の画面になる。

資料請求のボタンを押したあとこちらの画面に移動する。ここで顧客の情報と請求したい大学名と学部名を入力してもらう。

ここで入力してもらった情報はスプレッドシートの purchase シートに保存され、スプレッドシートの情報をもとにユーザーの住所に資料をお届けすることができる。

この画面をスライドすると申し込みするボタンがある。

そのボタンをタップすると下の画像のようにスプレッドシート側では、ユーザー情報が追加される仕組みになっている。



申し込みボタンをタップするとアプリ側では、申し込み完了ページに移動し、このようなテキスト画面が表示される。

「ホームへ戻る」をタップすると学校一覧のページに戻り、再度学校の情報を閲覧したり、資料請求がおこなえる事になる。