Universidad Nacional Autónoma de México

Facultad de Ciencias

PumaMóvil: Desarrollo de una aplicación para la Movilidad Urbana en la UNAM

T E S I S

que para obtener el título de:

Licenciado en Ciencias de la Computación

presenta:

Juan Antonio López Rivera

Tutor:

Dr. Carlos Gershenson García

Ciudad Universitaria, CD. MX., 2021

Índice general

Capítulo 1

Introducción

La congestión vial es uno de los principales problemas que enfrenta la Ciudad de México (CDMX) y la Zona Metropolitana del Valle de México (ZMVM). Un reporte global de movilidad de 2019 clasificó a la CDMX como la tercer ciudad más congestionada del mundo, y la sexta en términos de horas promedio perdidas al año por tráfico, con 158 horas [1]. Además de las horas perdidas de productividad, la pobre movilidad es un factor que empeora otros problemas que enfrenta la CDMX, como la inseguridad vial, siendo México el tercer país de Latinoamérica con el mayor número de muertes viales [2], y la emisión de contaminantes que perjudican la salud respiratoria, causantes de 48 mil muertes prematuras al año en el país [3].

La movilidad urbana depende de al menos 8 factores interrelacionados [4] entre ellos el horario de los viajes, la distancia, la cantidad de viajes, la capacidad de los sistemas de transporte y la infraestructura. La CDMX tiene problemas que afectan a varios de estos factores. Hay una alta concentración de servicios (educativos, laborales, de salud, etc.) en los distritos centrales de la ciudad, pero en éstas zonas el costo de vida suele ser bastante mayor que en las áreas periféricas de la ciudad o incluso los estados aledaños. Así que una proporción grande de la población vive lejos del lugar donde estudian o trabajan, y se desplazan diariamente aunque les

consuma horas adicionales. El resultado es una gran cantidad de viajes, muchos de ellos recorriendo grandes distancias, todos en la misma dirección (de la periferia al centro de la ciudad y viceversa), y en el mismo horario laboral o escolar, ocasionando congestiones diariamente en las "horas pico".

El problema de movilidad es multifactorial por lo que difícilmente habrá una solución única; se requieren implementar múltiples soluciones que lo enfrenten en diferentes aspectos y escalas [4]. Por un lado la planificación urbana juega un papel importante, y varios problemas pueden aliviarse construyendo y ampliando autopistas, carriles ciclistas, sistemas de transporte público, etc. Sin embargo la planificación depende de predicciones del comportamiento futuro de las ciudades, las cuales son limitadas – por ejemplo hace 100 años era imposible prever que el automóvil se volvería uno de los principales medios de transporte. Por este motivo son deseables soluciones complementarias que se adecúen mejor a la naturaleza dinámica de los problemas urbanos. Un grupo de soluciones tecnológicas propuestas bajo ésta premisa se conocen como *tecnología viviente* [5] debido a que es tecnología que presenta características de sistemas vivos, como adaptabilidad, robustez, y auto-organización. En particular la auto-organización, o coordinación descentralizada, es una propiedad muy útil que se ha aplicado para optimizar sistemas como rutas de transporte público [6] y la coordinación de semáforos [7].

Ésta tesis detalla el desarrollo de una aplicación multiplataforma (Web, iOS y Android) que busca mejorar la movilidad en la CDMX y la ZMVM facilitando la auto-organización de sus usuarios mediante sistemas de información y coordinación en tiempo real. Nuestro objetivo central es ofrecer a los usuarios un sistema de emparejamiento de viajes en tiempo real, para ayudarlos a organizar viajes compartidos en distintos medios de transporte. Nuestra hipótesis es que este sistema sería de utilidad para hacer más óptimos los viajes en automóvil, ya que en la ZMVM 68 % de los automóviles entre semana viajan sin pasajeros, y 22 % más viajan con solo un pasajero [8] – asientos vacíos que sin embargo consumen espacio y combustible.

Asimismo, la gran cantidad de viajes que se realizan en la ZMVM en los mismos horarios y en la misma dirección implica que hay muchos viajes potencialmente emparejables.

Aunque nuestro sistema de emparejamiento es generalizable y puede adaptarse para funcionar en distintas ciudades o comunidades, ésta aplicación está dirigida hacia miembros de la UNAM ubicados en la ZMVM, principalmente para visitantes de Ciudad Universitaria (CU). Se escogió ésta población por varios motivos: es suficientemente grande (en 2020 la UNAM contó 360 883 alumnos y 41 332 académicos), y una gran parte de ésta población realiza viajes alrededor de los horarios escolares de entrada y salida, con un plantel de origen o destino en común, aumentando las posibilidades de emparejamiento. Pero otra cuestión importante es la seguridad, pues una de las mayores barreras que identifican los usuarios al considerar usar un sistema de 'ridesharing' es la inseguridad de compartir viajes con desconocidos. Sin embargo estudios muestran que pertenecer a la misma comunidad incrementa la confianza mutua de los usuarios [9]. Por ello al registrarse en la aplicación nuestros usuarios deben proporcionar datos para validar su registro activo en la UNAM, lo que nos permite adicionalmente conservar un registro validado de datos de cada usuario. Para mejorar la percepción de seguridad de usuarios, antes de aceptar un viaje se puede consultar información mutua como el plantel, nombre y foto de perfil. Además ofrecemos funcionalidades de seguridad complementarias, entre ellas un botón de emergencia para llamar de inmediato a la seguridad local, una opción para compartir su ubicación en tiempo real con un contacto de elección, y recursos informativos sobre la ubicación de elementos de seguridad en C.U.

Tal sistema de emparejamiento de viajes es similar a una red social en el sentido de que su funcionamiento depende totalmente de la actividad de los usuarios; es necesario alcanzar una *masa crítica* de usuarios realizando viajes emparejables, de lo contrario los usuarios nuevos que intenten emparejar viajes no encontrarán ofertas y eventualmente dejarán de intentarlo. Por este motivo se implementaron

funcionalidades relativas a movilidad con el objetivo de atraer y retener usuarios. Gracias a una colaboración con la DGSGM se instalaron localizadores GPS en los Pumabuses de C.U., lo que permite consultar en la app la ubicación de los Pumabuses más cercanos según su ruta. También se integró un sistema de conteo en módulos de Bicipuma, con el cual podemos mostrar en cada módulo el número de bicicletas disponibles. La aplicación también incluye mapas estáticos del Metro y Metrobus de la CDMX, y una sección informativa de actividades y eventos nutrida por Gaceta Digital UNAM.

Ésta tesis está estructurada de la siguiente manera. El capítulo 2 abarca el marco teórico, y explica los modelos que se han propuesto para el estudio de ciudades, incluyendo el paradigma actual que son las ciencias de la complejidad, así como algunas propiedades y aplicaciones de la complejidad en ciudades, y las características específicas del problema de movilidad en la CDMX y ZMVM. El capítulo 3 describe el funcionamiento de cada sistema de la aplicación desarrollada. EL capítulo 4 detalla las contribuciones particulares del autor de ésta tesis al desarrollo del proyecto. Finalmente en el capítulo 5 hay una discusión de resultados y evaluación del software bajo métricas estándar de calidad, con una breve conclusión en el capítulo 6.

Capítulo 2

Marco teórico

Para poder proponer, sustentar y evaluar soluciones de movilidad en áreas urbanas, es de utilidad entender lo mejor posible el funcionamiento intrínseco de éstos sistemas. Por ello es relevante esclarecer una pregunta fundamental: ¿Qué son las ciudades? O formalmente: ¿Qué modelo las representa mejor? ¿Cuales son sus propiedades y aplicaciones?

Éste capítulo está estructurado de la siguiente manera. Primeramente hay un resumen de la teoría de los sistemas complejos o ciencias de la complejidad, la cual se ha desarrollado en las últimas décadas como modelo de una multitud de sistemas naturales, y es la base de la solución plasmada en la aplicación desarrollada. Después se describe brevemente el planteamiento de los principales modelos que han sido propuestos para describir áreas urbanas – sistemas mecánicos, organismos vivos, y sistemas complejos – incluyendo algunos de los principales resultados y limitaciones que presenta cada teoría. Finalmente se incluye una caracterización del problema de movilidad que existe en la Ciudad de México y la Zona Metropolitana del Valle de México.

2.1. Los sistemas complejos

La palabra *complejidad* proviene del latín *plexus* que significa entrelazar. Esto ilustra un principio fundamental del planteamiento de los sistemas complejos: están formado por componentes profundamente interconectados o entrelazados, por lo que un cambio en un componente se propaga hacia otros mediante una red de interacciones, incluso volviendo a afectar al componente inicial [10].

Este planteamiento es opuesto a la teoría reduccionista y determinista que predominaba en la ciencia antes del inicio del siglo XX, que planteaba que para comprender completamente un sistema hay que estudiar de manera aislada sus componentes para hallar sus propiedades fundamentales, y que para cada sistema existen una serie de principios que permiten predecir absolutamente su evolución futura a partir de cualquier estado. Éstas suposiciones quizás son razonables para sistemas simples, pero casi todos los fenómenos que ocurren en la realidad son complejos.

Los modelos reduccionistas de sistemas complejos tienen una capacidad predictiva limitada porque ignoran las interacciones entre elementos: de estas interacciones emergen propiedades nuevas que no existían en los elementos individuales, y estas propiedades afectan al sistema en distintas escalas y generan así mismo nuevas propiedades; de esta manera eventualmente estos sistemas se alejan de cualquier conjunto de reglas estáticas que se propongan. Estos sistemas presentan un comportamiento caótico y no determinista, donde condiciones iniciales ligeramente distintas pueden ocasionar resultados ampliamente diferentes.

Otra consecuencia de que todos los elementos estén interconectados es que cualquier perturbación individual se propaga en distintas escalas y puede cambiar significativamente el desarrollo del sistema, Esto implica que la única manera de conocer precisamente el futuro estado y comportamiento de un sistema complejo es simular en su totalidad sus componentes e interacciones. Ésta limitación es equivalente al principio de irreducibilidad computacional [11], que dice que en general la única for-

ma de predecir el resultado de un cómputo es realizándolo, ya que algunas preguntas son indecidibles y no existen atajos lógicos o matemáticos hacia su respuesta.

Se han propuesto distintas formas de medir la complejidad en varios contextos, incluyendo el computacional, social, económico, y biológico [12]; sin embargo no existe una medida universal de complejidad. Una definición general y recursiva es que la complejidad de un sistema aumenta en relación con el número de componentes distintos, el número de conexiones entre ellos, la complejidad de los componentes, y la complejidad de las interacciones [13].

A continuación se describen brevemente las principales propiedades de los sistemas complejos.

Auto-organización

Intuitivamente podemos definir los sistemas auto-organizantes como aquellos que producen un patrón global a partir de las interacciones de sus componentes. Algunos ejemplos típicos son enjambres de insectos, parvadas de pájaros, y cardúmenes de peces. Independientemente del mecanismo que ocupan, en éste tipo de sistemas su orden aumenta conforme transcurre el tiempo como resultado de sus propias dinámicas internas.

Algunos autores argumentan que la caracterización de sistemas como "ordenados" (o incluso "complejos") es subjetiva, pues depende del propósito del observador y el significado que le asigna a los estados más probables del sistema. En general cualquier sistema dinámico puede describirse como auto-organizante pues presentan atractores hacia los cuales tienden los elementos [14]. Los sistemas complejos son un caso peculiar de auto-organización, ya que dependiendo de la escala y medida usada un mismo sistema puede parecer tanto auto-organizante como auto-desorganizante [13]. Esto ocurre porque éstos sistemas contienen tanto dinámicas que tienden al orden como dinámicas aleatorias o de ruido blanco.

Propiedades emergentes

El concepto de emergencia ha sido estudiado por siglos. Se refiere a propiedades de un sistema que son visibles en una escala grande pero no están presentes a baja escala. Por ejemplo el color, conductividad, y maleabilidad del oro son propiedades que no existen en los átomos de oro solos.

Éste y varios conceptos relacionados con sistemas complejos han sido difíciles de definir concretamente, porque han sido usados en múltiples contextos bajo distintos significados. En particular muchas veces no se hace distinción entre propiedades emergentes y la auto-organización. Algunas formalizaciones basadas en la teoría de información de Shannon definen la emergencia como la información novedosa generada por un sistema [15], o un incremento en la información generada comparada con la información inicial o de entrada. Bajo la teoría de Shannon los sistemas ordenados contienen menos información que los sistemas aleatorios, así mientras que la auto-organización implica un incremento de orden y reducción de la información, la emergencia es lo opuesto ya que implica un aumento de la entropía del sistema y mayor información.

Homeostasis (robustez y adaptabilidad)

La homeostasis se refiere a la capacidad de un organismo o sistema de mantener un estado estable de funcionamiento a pesar de cambios internos y externos. Se puede ver como una resistencia al cambio por parte de un sistema cuando ya se encuentra en condiciones óptimas. Ésto no implica que los sistemas en homeostasis tienen un único punto de equilibrio, en realidad la homeostasis existe en un rango de valores deseables para ciertas variables del sistema.

Para lograr la homeostasis los sistemas emplean mecanismos de control que se conforman por al menos tres componentes interdependientes: un receptor o sensor, un centro de control, y un efector que es la variable que se desea regular. En sistemas

complejos usualmente hay múltiples variables que se desean mantener entre ciertos valores críticos, y varios sistemas interconectados que efectúan distintos grados de control sobre cada variable. Por ejemplo en los organismos vivos algunas variables esenciales incluyen la temperatura interna, la presión arterial, y el pH extracelular, mientras que otras variables fluctúan con mayor libertad como mecanismo de control, por ejemplo el ritmo cardiaco para modular la presión arterial [16].

Desde el punto de vista de la teoría de la información, una alta homeostasis implica que no ocurren cambios en el sistema y la cantidad de información se mantiene igual [15]. Ésto ocurre naturalmente en sistemas altamente ordenados, pero también es una propiedad deseable en sistemas complejos. En éstos sistemas la homeostasis implica una alta *robustez* en un sistema, pero también *adaptabilidad* ante condiciones externas. Los sistemas complejos son altamente robustos y adaptables porque existen en un estado que no es totalmente ordenado ni desordenado, sino el punto de transición entre ambos. Uno de los descubrimientos que desató inicialmente el estudio de sistemas complejos es que estos sistemas presentan lo que nombraron "criticalidad auto-organizada" (Self Organized Criticality) [17], existen en un punto crítico entre dinámicas ordenadas y aleatorias, lo que les permite beneficiarse de propiedades de ambas como se explica a continuación.

Criticalidad

La existencia de puntos de transición de fase fue descubierta a inicios del siglo XIX, cuando físicos que estudiaban la transición entre el estado líquido y gaseoso del agua hallaron que justo en el punto de transición entre ambos el agua presenta propiedades inusuales; cerca de éste punto al que nombraron *punto crítico*, el líquido se comporta como vapor, por ejemplo se vuelve compresible y mal conductor eléctrico, y el vapor como líquido, y exactamente en el punto crítico el agua existe en una única fase que comparte propiedades de tanto líquido como vapor.

Tiempo después en las décadas de 1960 y posteriores se mostró que fenómenos

invariantes de escala como fractales y leyes de potencias emergen en el punto crítico entre fases [18]. Estas propiedades de puntos críticos – fractalidad, leyes de potencias, y ruido rosa (también llamado ruido 1/f) – se encontraron en múltiples sistemas complejos naturales, pero fue hasta la década de 1980 que se encontró un modelo que demuestra cómo surgen espontáneamente propiedades críticas en la naturaleza, cuando se mostró que algunos autómatas celulares presentan comportamiento complejo (figura 2.1 d) [19, 20].

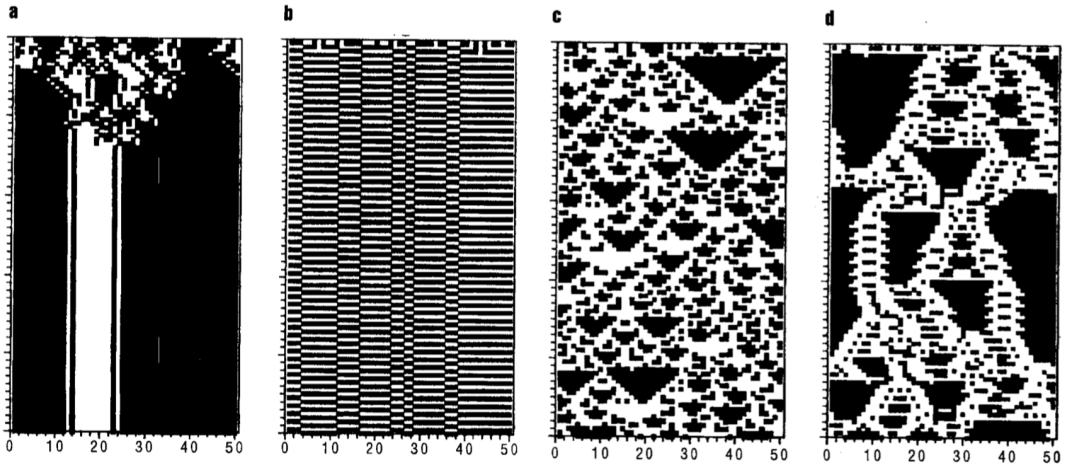

Figura 2.1: Los distintos tipos de comportamiento que presentan los autómatas celulares: (a) constante, (b) periódico, (c) caótico, y (d) complejo. Fuente: Vicente Solé R. et al. 1995.

Los autómatas celulares son modelos computacionales simples que usualmente consisten en una cuadrícula, donde cada espacio puede tomar un valor binario (negro/blanco), y el valor de cada espacio se determina por unas pocas reglas relativas al estado de los espacios adyacentes. Dependiendo del conjunto de reglas seleccionado, los autómatas celulares pueden producir patrones constantes, periódicos, complejos y fractales, y su estudio ilustra como patrones globales complejos (con ruido rosa y auto-similitud fractal) pueden surgir espontáneamente a partir de interacciones locales simples.

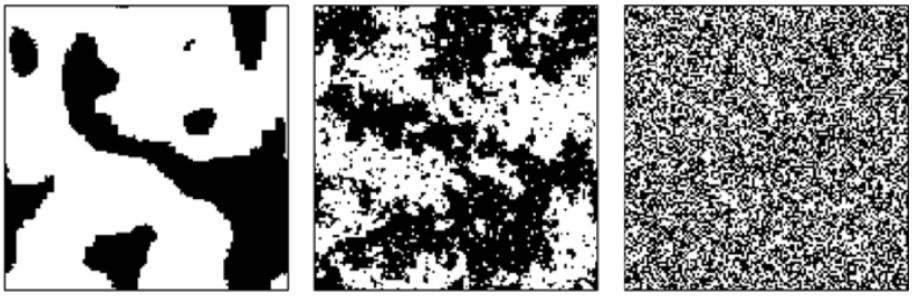

Figura 2.2: El modelo de Ising es un ejemplo común de fractalidad espacial (imagen en medio) emergente en el punto de transición entre dinámicas ordenadas (izquierda) y aleatorias o de ruido blanco (derecha). Similarmente a los autómatas celulares, en este modelo el estado de cada espacio es binario y se determina por sus vecinos inmediatos, pero existe una variable global que controla la facilidad de una casilla de cambiar de estado, permitiendo simular distintos grados de entropía. Fuente: Kitzbichler M.G. et al. 2009.

Los sistemas complejos buscan existir en criticalidad porque entre otras cosas es el punto óptimo para la trasmisión de información [21], ya que la distancia promedio entre elementos suele ser muy pequeña comparada con otras configuraciones. Intuitivamente, una perturbación en un sistema bajo dinámicas aleatorias no se propagará muy lejos pues hay demasiado ruido, mientras que un sistema muy ordenado es rígido y las perturbaciones o casi no se propagan o afectan demasiado a todo el sistema, como una onda sonora en un sólido. Al existir cerca de un punto crítico los sistemas complejos pueden beneficiarse de tanto homeostasis y auto-organización como de emergencia y adaptabilidad.

2.2. Paradigmas para modelar áreas urbanas

2.2.1. Las ciudades como sistemas mecánicos

Fue hace más de medio siglo que las ciudades fueron planteadas por primera vez como *sistemas* [22] definidos como colecciones distintas de entidades que interactúan

entre ellas, usualmente en equilibrio, y cuyo funcionamiento es controlable por un subsistema central que recibe retroalimentación negativa y mantiene o restaura tal equilibrio mediante planificación y organización.

Se encontraron muchas insuficiencias bajo ésta teoría [23]. Se mostró que no existe un único estado de equilibrio, las ciudades no suelen regresar a estados previos de equilibrio ya que están cambiando y creciendo constantemente, no son sistemas cerrados pues reaccionan a cambios externos en el ambiente, y no están organizadas centralmente, sino que evolucionan principalmente como resultado de millones de decisiones individuales y grupales, solo ocasionalmente se toman acciones por parte de un organismo central.

Más indicios de que los sistemas mecánicos son inadecuados para representar ciudades fueron surgiendo conforme progresó el estudio del crecimiento urbano. En 1970 se mostró que en ciudades reales el cambio no es gradual ni continuo como asumían modelos previos basados en ecuaciones diferenciales, sino que se da en saltos repentinos debido a ciclos de retroalimentación positiva que ocasionan crecimiento (y decrecimiento) logístico [24] con oscilaciones mientras los sistemas establecen un nuevo equilibrio [25]. Asímismo se mostró que las áreas urbanas presentan comportamientos espaciales caóticos, pues resultan cualitativamente distintos aún partiendo de condiciones iniciales casi idénticas [26].

Era clara la necesidad de modelos dinámicos que permitan explicar y simular escenarios de crecimiento muy distintos entre sí. Conforme se hallaban más limitaciones en los modelos reduccionistas, fue adquiriendo mayor relevancia el estudio de las semejanzas entre ciudades y seres vivos.

2.2.2. Las ciudades como sistemas orgánicos

Semejanzas entre las áreas urbanas y los sistemas vivos han sido notadas por múltiples autores en diversas áreas [23, 27, 28]. Las ciudades tienen una organi-

zación interna semejante a un metabolismo, con redes de transporte de materia, energía e información; las telecomunicaciones en particular han sido caracterizadas como "sistemas nerviosos" [29]. También crecen y se reparan a sí mismas, aunque sus mecanismos para ello son más cercanos a los de grasas que a los de animales [4]. Incluso desde la perspectiva termodinámica, las ciudades obtienen materia, energía e información de su ambiente, los transforman, y producen desechos para mantener su organización. La pregunta entonces es si éstas similitudes implican la existencia de principios subyacentes, cuantitativos y predictivos, que son los mismos en tanto sistemas urbanos como vivos, o si son solamente similitudes cualitativas y hay diferencias fundamentales entre ellos.

Recientemente un estudio sobre datos de múltiples ciudades mostró que sí existen diferencias fundamentales entre ciudades y seres vivos, al menos en ciertas propiedades de ciudades que pertenecen a clases universales que no existen en organismos vivos [30]. Compararon el tamaño de ciertos componentes de ciudades con el tamaño total del sistema, planteando todas las relaciones de crecimiento como una ley de potencia de la forma: $Y(t) = Y_0 N(t)^{\beta}$ (donde Y(t) es una propiedad que se desea estudiar, N(t) es la población total como una medida del tamaño del sistema, Y_0 una constante de normalización y β un exponente de crecimiento). De esta manera hallaron que todas las propiedades urbanas que estudiaron se agrupan en alguno de tres distintos valores de β, algunas con $\beta \approx 0.8 < 1$, otras tienen $\beta \approx 1$, y en las demás $\beta \approx 1.1 - 1.3 > 1$. [1] En contraste, estudios de biología desde 1932 han hallado

[1] Las propiedades con $\beta \approx 0.8 < 1$ son economías de escala y se relacionan con infraestructura, incluyen el área total pavimentada, la longitud de líneas eléctricas, y el número de gasolineras. Las que tienen $\beta \approx 1$ usualmente están directamente asociadas a necesidades de seres humanos individuales como el número de viviendas, de empleos, el consumo eléctrico y de agua. Lo más notable es que una gran cantidad de propiedades urbanas escalan superlinealmente, con $\beta \approx 1.1 - 1.3 > 1$, es decir su rendimiento incrementa conforme aumenta el tamaño de la población. Las propiedades en ésta clase son relativas a interacciones sociales que resultan en procesos de innovación y de creación de riqueza, como el Producto Interno Bruto, el número de patentes nuevas, y sueldos totales, pero también incluye propiedades como índices de crimen, la transmisión de enfermedades infecciosas como el SIDA, y incluso la velocidad de caminata de peatones.

que a pesar de la enorme complejidad y diversidad de formas de vida existentes, casi todas las características fisiológicas de organismos vivos escalan como función de la masa corporal M, también bajo una ley de potencia pero su exponente usualmente es múltiplo de $1/4$ y menor a 1 [31, 32]. Por ejemplo la tasa metabólica R (la energía ocupada por un organismo) escala bajo la relación $R \propto M^{3/4}$. Ésto mostró que las ciudades son sistemas fundamentalmente distintos de los seres vivos, pero también el hecho de que los valores de β se aglomeren alrededor de tres valores indica que muchos fenómenos urbanos que aparentan ser distintos entre sí obedecen las mismas dinámicas subyacentes.

El motivo de las semejanzas entre ciudades y seres vivos es que ambos son sistemas complejos. En estos sistemas las decisiones individuales e interacciones mutuas de grandes cantidades de individuos, ya sean células o personas, causan que el sistema completo adquiera una organización robusta, así como globalmente surgen nuevas propiedades y patrones conforme se optimizan las interacciones entre individuos. Por ejemplo, la presencia universal en la biología del escalamiento bajo ley de potencias se ha explicado como una manifestación de principios subyacentes que restringen las dinámicas y geometría de las redes de distribución de recursos en organismos [33]. Los sistemas vivos requieren una integración cercana y eficiente de cantidades enormes de unidades, y para lograr esto la vida en todas las escalas está sustentada por redes jerárquicas con formas fractales, ya que éstos patrones ocupan espacio óptimamente [34]. La estructura de dichas redes determina y limita la velocidad de transferencia de recursos en todos los organismos vivos, por lo que una multitud de propiedades de un organismo escalan como función de su tamaño. Lo mismo ocurre en ciudades, y su crecimiento y morfología se puede entender como resultado de la estructura y optimización de sus redes subyacentes de distribución de recursos [30].

A fines del siglo pasado fue creciendo el interés en el estudio de los llamados sistemas complejos, que incluyen no solo ciudades y organismos, sino todo tipo de

sistemas constituidos por enormes cantidades de unidades individuales que interaccionan mutuamente, desde colonias de insectos hasta sistemas climáticos. Éstos sistemas presentan propiedades intrigantes que no habían sido posibles de explicar formalmente; por ejemplo éstos sistemas son bastante robustos, se adaptan y sobreviven ante cambios externos e internos, en contraste con la mayoría de sistemas mecánicos construidos por humanos, que suelen ser frágiles y pueden colapsar si falla el funcionamiento de tan solo un componente. Ésta linea de investigación expuso nuevamente errores fundamentales en el planteamiento científico reduccionista que predominaba hasta inicios del siglo XX – en particular se cometía el error de ignorar las interacciones entre partes de un sistema para poder estudiar cada componente de forma aislada, pero las interacciones son también un componente esencial de éstos sistemas, y su estudio ha resultado en la teoría incipiente de las ciencias de la complejidad.

2.2.3. Las ciudades como sistemas complejos

Fractalidad

La morfología de las ciudades se determina por distintos procesos de decisiones espaciales en los cuales individuos y grupos se localizan de acuerdo a sus ubicaciones mutuas y las actividades posibles según el contexto, que se define por el uso de suelo. Esto resulta en aglomeraciones alrededor de subcentros o *clusters* de distintas escalas organizados alrededor de ciertas actividades económicas cruciales, pues ocurre un balance de *trade-offs* o ventajas y desventajas de acuerdo a la distancia relativa entre distintos espacios [35].

Bajo estos procesos las áreas urbanas tienden a estructurarse de manera fractal en el espacio, pues se ha mostrado que su forma tiene una auto-similitud estadística o jerarquía de clusters [36] que sigue una ley de potencias. Asímismo se ha encontrado que el crecimiento urbano es fractal [35]. En consecuencia se pueden aplicar

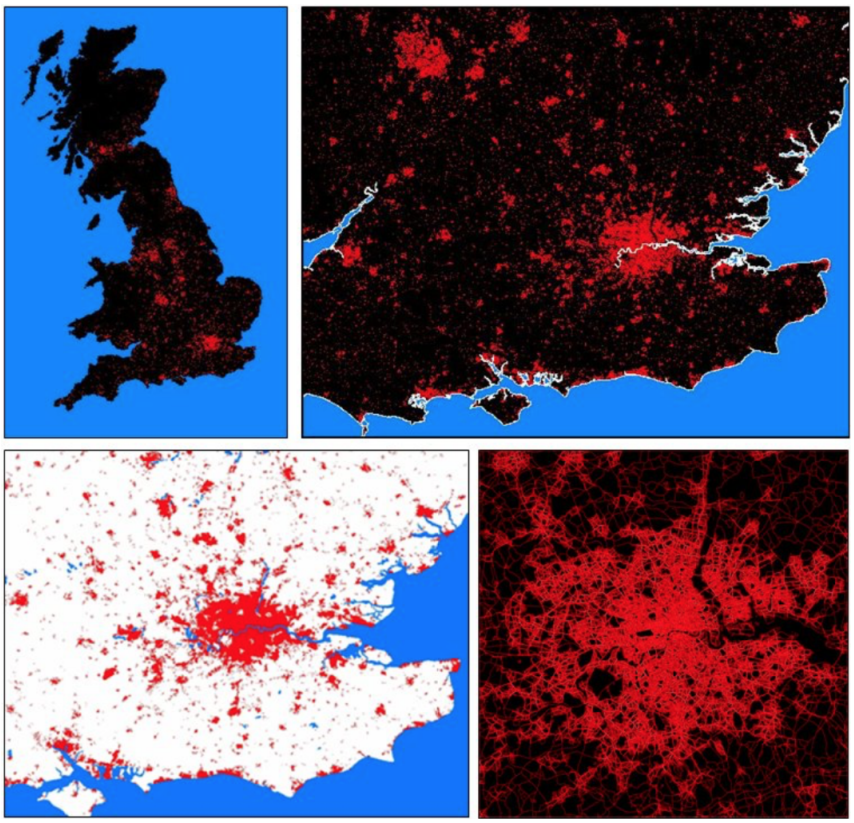

Figura 2.3: La distribución de población en el Reino Unido y Londres (ambas imágenes de arriba, y abajo a la izquierda) tiene un comportamiento auto-similar en distintas escalas, también aparente en la red vial de la ciudad de Londres (abajo a la derecha). Fuente: Batty M., 2011.

herramientas de geometría fractal para estudiar diferentes patrones y procesos urbanos, por ejemplo la dimensión fractal se ha usado para clasificar ciudades por su densidad [35], y se ha mostrado que la dimensión fractal tiende a crecer como función del tiempo [37].

Leyes de potencia

La presencia de leyes de potencia en múltiples factores urbanos ha sido estudiada a lo largo del siglo pasado. Uno de los primeros ejemplos es la Ley de Zipf, que

halla una relación inversa en la distribución rango-frecuencia de las poblaciones de ciudades – es decir la n-ésima ciudad más poblada tiene 1/n la cantidad de habitantes de la que está en primer lugar. Otros fenómenos urbanos que presentan esta relación inversa en la distribución rango-frecuencia incluyen el tamaño de negocios, el capital de individuos, y la longitud de viajes realizados. También está presente en sus aspectos dinámicos, pues los patrones de flujos migratorios entre ciudades se han modelado como fuerzas gravitatorias [38] – el efecto disuasivo de la distancia al tomar decisiones sobre migración es análogo a la disminución de la fuerza de gravedad bajo un cuadrado inverso, que es una ley de potencia.

Una explicación común de cómo surgen las leyes de potencia es el proceso de "apego preferencial" (*preferential attachment*) también conocido como "los ricos se hacen más ricos", en el cual un recurso o riqueza se distribuye entre individuos como función de la cantidad que ya poseen, de manera que conforme más recursos tengan más recursos obtienen. Éste proceso genera distribuciones con colas largas, que se pueden describir como leyes de potencia o distribuciones de Pareto (conocida popularmente como la regla de 80/20).

Redes libres de escala y de mundo pequeño

La ciencia de redes ha proporcionado diversas herramientas útiles para modelar y simular procesos en sistemas complejos, entre ellos ciudades. En 1999 se mostró que aplicar el proceso de apego preferencial en la generación de redes aleatorias – agregando nuevas conexiones entre nodos bajo una probabilidad que es mayor conforme más conexiones tenga un nodo – resulta en redes que asemejan la estructura de muchos fenómenos sociales y naturales, incluyendo las redes de citas de artículos científicos, las interacciones entre proteínas, y el Internet [39]; a éstas se les nombró *redes libres de escala*.

Estas redes presentan aglomeraciones en comunidades o "clusters" alrededor de "hubs" o nodos centrales, donde el tamaño de los clusters y el número de conexiones

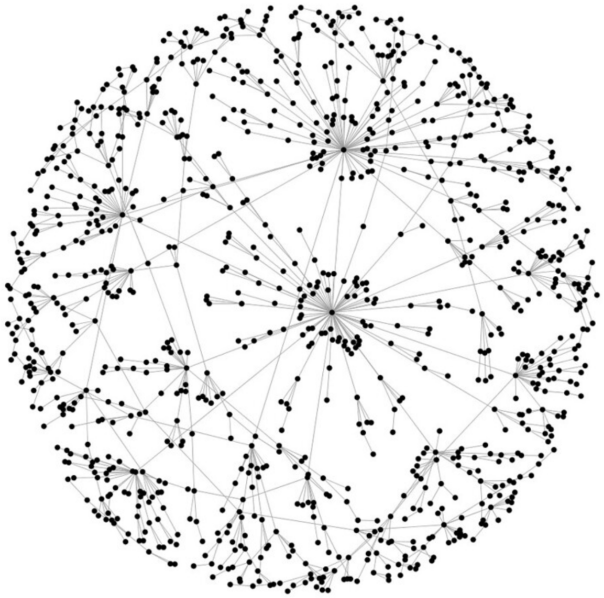

Figura 2.4: Una red libre de escala dispersa del tipo Barabasi-Albert, n = 1000. Fuente: Simon Cockell, 2013.

por nodo siguen una ley de potencia, es decir existen unas pocas aglomeraciones enormes con una gran cantidad de nodos interconectados, y muchas comunidades periféricas pequeñas. Estas redes se organizan de forma jerárquica: los clusters grandes están conectados con unos de menor tamaño, los cuales se conectan con otros menores y así sucesivamente. De esta manera si un nodo falla hay baja probabilidad de que sea un hub esencial. Otro beneficio de estas redes es que usualmente la distancia entre nodos es pequeña comparada con estructuras altamente ordenadas como por ejemplo una cuadrícula.

Recientemente ha habido debate sobre si las distribuciones con colas largas que han sido observadas en la naturaleza siguen leyes de potencia o en realidad siguen una distribución *lognormal* [40], generada por un proceso de crecimiento logístico descrito por la "ley de Gilbrat". La dificultad es que ambas distribuciones son muy similares, y en la práctica se necesitan grandes cantidades de datos para distinguirlas

ya que matemáticamente solo son distintas en su límite [41].

Otro tipo de redes que surge frecuentemente en la naturaleza son las *redes de mundo pequeño*, nombradas así porque en ellas la distancia promedio entre nodos L es muy pequeña, suele ser proporcional al logaritmo del número de nodos totales N, esto es $L \propto logN$. Estas redes tienen un alto coeficiente de clustering [42]. Ejemplos de ellas incluyen la red de colaboración entre actores de cine, la red neural del gusano *Caenorhabditis elegans*, y la red eléctrica de los Estados Unidos [43]. Las redes libres de escala no suelen ser de mundo pequeño, y solo algunas redes urbanas como la red de transporte aéreo son inequívocamente de mundo pequeño, mientras que otras como el tránsito de autobuses y las redes sociales asemejan mundo pequeño solo bajo algunas medidas [44].

Uno de los mayores desafíos actuales es describir cómo se acoplan y interactúan las distintas redes de un sistema complejo. En ciudades por ejemplo existen una multitud de redes que se sobrelapan y funcionan simultáneamente, algunas que dependen fuertemente del componente espacial como las redes de distribución de recursos y de movilidad, pero también existen redes sociales cuyo relación con el espacio físico es indirecta.

2.2.4. Aplicaciones de la auto-organización en ciudades

La capacidad de los sistemas complejos de auto organizarse ha inspirado una multitud de sistemas logísticos artificiales [4, 45] que basados en los principios de modularidad y coordinación descentralizada buscan solventar problemas regulando las interacciones de potencia entre elementos, en lugar de diseñar soluciones directamente. Éstas soluciones son útiles para problemas no estacionarios, abundantes en sistemas complejos, pues facilitan la capacidad de los sistemas de adaptarse ante cambios dinámicos en la situación. A continuación se mencionan algunas soluciones de este tipo que se han aplicado hacia problemáticas de movilidad en áreas urbanas.

Coordinación de semáforos

El tráfico es un gran ejemplo de una demanda dinámica e impredecible, donde la situación cambia constantemente en cuestión de segundos. Ocasiona varios problemas no triviales, por ejemplo la coordinación de semáforos, que se ha mostrado es un problema EXP-completo [46]. Las soluciones tradicionales buscan optimizar la duración de luz verde que maximiza el movimiento promedio de vehículos, para llegar a una única duración fija. Pero la cantidad de vehículos esperando en un semáforo varía en cada instante, por lo que es deseable que la duración de cada semáforo se adapte y cambie para minimizar el tiempo muerto. Otro método tradicional involucra coordinar las luces verdes consecutivas para que los vehículos no tengan que detenerse, esto se conoce como "ola verde" (*green wave method*). Esta solución falla cuando la velocidad de los vehículos es más lenta o más rápida que la velocidad de la ola de luces verdes, y tampoco es óptima para vehículos que fluyen en dirección opuesta a la ola [47]. Las soluciones auto-organizantes plantean reglas para que las intersecciones determinen, independientemente unas de otras, cuándo cambiar de luz basándose solamente en información de tráfico inmediata a la intersección, como la cantidad de vehículos esperando y el tiempo que llevan detenidos. Se ha mostrado que las soluciones auto-organizantes son más robustas y adaptables que soluciones de optimización fijas [47], y su implementación ha resultado en mejoras considerables en los tiempos de espera de autos, peatones y transporte público [48].

Vehículos de transporte público

Otro problema de movilidad es la coordinación de vehículos de transporte público sobre una misma ruta. En los últimos 50 años los esfuerzos de teoría de transporte se habían enfocado en diseñar sistemas para mantener los vehículos equidistantes entre sí (*equal headways*) [49], porque en teoría minimiza el tiempo de espera promedio de los pasajeros en estaciones. Sin embargo un sistema forzado de vehículos

equidistantes requiere que algunos vehículos desperdicien tiempo en estaciones, y es inestable ya que demoras de vehículos individuales se amplifican y suelen colapsar la configuración formando pelotones [45]. Recientemente se ha propuesto un método auto-organizante inspirado en la comunicación de hormigas por medio de feromonas, pero éste método ocupa *antiferomonas* [4], marcadores virtuales en las estaciones que incrementan su concentración linealmente con el paso del tiempo y son borrados cuando un vehículo pasa por encima. Un algoritmo computacional sencillo determina cuanto tiempo debe esperar un vehículo en una estación dependiendo del número de pasajeros en la estación, la concentración de antiferomonas adelante, y la distancia al vehículo de atrás. Sorprendentemente éste método tiene un desempeño supra-óptimo, pues resulta en tiempos de viaje menores que incluso el óptimo teórico de vehículos equidistantes. Esto es porque para mantenerse equidistantes los vehículos deben desperdiciar tiempo en estaciones con pocos pasajeros y ignorar a algunos de estaciones demasiado llenas. La configuración auto-organizante no colapsa bajo perturbaciones causadas por demoras, ya que está diseñada para adaptarse a la demanda.

Información en tiempo real

Los avances tecnológicos han permitido crear sistemas de información en tiempo real que ayudan a los usuarios a adaptar sus rutas según la situación actual de tráfico, facilitando así la auto-organización del sistema. Un ejemplo que ha existido por décadas es la información de tráfico en tiempo real trasmitida por estaciones de radio. Ésta tiene la limitación de que todos los conductores reciben la misma información, pero una gran parte de ella no les será relevante y no se puede solicitar información particular. En el siglo XXI la comercialización de dispositivos con GPS permitió a los conductores consultar la ruta más corta hacia su destino – pero la ruta más corta no necesariamente es la más rápida, ya que cuando mucha gente adopta el uso de GPS, las rutas más cortas se saturan. Éste efecto se ha solucionado en los últimos

años con aplicaciones como Google Maps y Waze que ofrecen información de tráfico y accidentes en tiempo real basados en localización. Aprovechan la cantidad de usuarios activos para usar sus dispositivos como sensores distribuidos y así detectar bloqueos en el flujo vehicular, para informar en tiempo real a los usuarios sobre rutas alternas según su ubicación [4].

Asímismo la información en tiempo real de sistemas de transporte público puede ayudar a los pasajeros a planificar sus rutas más eficientemente [50]. Otra aplicación posible es adaptarse a estados dinámicos de tráfico mediante comunicación intravehicular [51]. En general hay un amplio potencial de aplicaciones de servicios basados en localización [52].

Mediadores pasivos

Los medios para incentivar la auto-organización no necesariamente tienen que ser tecnológicos. Una estudio piloto reciente en el Metro de la CDMX [53] muestra cómo una intervención "simple" puede implementarse para incentivar la auto-organización, en este caso para optimizar el abordaje de trenes. Los usuarios de sistemas como el metro de la CDMX normalmente siguen una estrategia egoísta para abordar, lo que en estaciones saturadas puede ocasionar empujones y no permitir que gente salga del tren, colapsando los sistemas. La intervención del estudio piloto consistió en letreros claros para señalizar en dónde estarán las puertas del vagón y indicando cómo dejar salir a los usuarios (figura). Los usuarios adoptaron positivamente el sistema; se esperaban aglomeraciones a ambos lados de las puertas, pero espontáneamente los usuarios formaron filas. Éstas en ocasiones se extienden hasta las escaleras y el piso superior, y han resultado en tiempos de abordaje reducidos y en eliminar casi completamente incidencias de conflictos y empujones. Ésta dinámica de abordaje se diseminó naturalmente y se volvió una norma aceptada.

Ésta intervención logró cambiar el comportamiento colectivo de los usuarios implementando un *mediador* sobre las interacciones entre personas. Es un ejemplo

Figura 2.5: La señalización instalada para mediar el ascenso y descenso de pasajeros en la estación Balderas del Metro de la CDMX. El éxito del piloto llevó a que se expandiera éste proyecto hacia varias otras estaciones de alta demanda. Fuente: Carreón et al., 2017.

de un sistema de control *pasivo*, donde las interacciones se regulan simplemente ofreciendo información útil.

Viajes compartidos

Los sistemas de viajes compartidos (*ridesharing*) son un gran ejemplo de una solución auto-organizante a problemas de movilidad. Son sistemas de transporte en los cuales dos o más viajeros individuales que van a realizar viajes con horarios y trayectos similares se coordinan para compartir un mismo vehículo (en el caso de automóviles se conoce como *carpooling*), resultando en ahorros de costos de gasolina, casetas y estacionamientos para los viajeros, así como globalmente reduce el número de vehículos en tránsito, mejorando la circulación y disminuyendo la cantidad de contaminantes emitida [54]. Algunos consideran los sistemas de transporte como taxis y Uber un tipo de *ridesharing* pero no son relevantes para ésta discusión, pues éstos no tienen los beneficios que resultan de juntar personas que iban a hacer el mismo viaje en un solo vehículo, los conductores transitan con el único propósito de transportar a los usuarios.

En el último siglo ha habido múltiples implementaciones de sistemas de viajes compartidos. En la década de 1940 en Estados Unidos surgieron clubes para organizar automóviles compartidos, incentivados por el gobierno con el fin de conservar

recursos para la guerra. En éstos sistemas los emparejamientos se organizaban mediante tableros de anuncios ubicados en los sitios de trabajo. Posteriormente en la crisis energética de la década de 1970 volvieron a surgir proyectos para conservar gasolina incentivados por empresas y el gobierno, de nuevo organizados mediante tableros de anuncios, pero surgen también nuevas propuestas, como *vanpooling* con camionetas subsidiadas y carriles de autopistas designados para vehículos con muchos ocupantes [55]. En las décadas de 1980 y 1990 disminuyó la urgencia de proyectos de conservación de energía, y los esfuerzos de transporte se tornaron hacia los problemas de congestión y calidad de aire. En éste periodo surgen las primeras implementaciones de sistemas tecnológicos de viajes compartidos basados en operadores telefónicos y anuncios por Internet. Pero cuando los precios de gasolina bajaron drásticamente, estos sistemas fueron perdiendo competitividad y cayeron en desuso. Sin embargo éstos primeros intentos son la base de muchos servicios modernos de *ridesharing*.

Es posible identificar al menos tres distintas poblaciones de usuarios de potencia de un sistema de viajes compartidos:

- **Larga distancia**. Los viajes de larga distancia o interestatales usualmente son planeados con días de anticipación y suelen ser flexibles en sus requerimientos de tiempo y lugar de encuentro, por lo que son ideales para la organización de viajes emparejados. 'BlaBlaCar' es un ejemplo de una aplicación para éste tipo de viajes compartidos que ha alcanzado una masa crítica de usuarios en México.

- **Viajes diarios**. Se refiere a los viajes recurrentes de ida y vuelta hacia escuelas y trabajos. Han surgido muchos sistemas de viajes compartidos para viajeros de una misma empresa o escuela, porque emparejar sus viajes tiene varias ventajas: una vez que se forman parejas o grupos se pueden repetir diariamente, y hallar coincidencias es relativamente fácil porque en muchos viajes

dos de las tres variables son iguales, horario y destino (u origen si es un viaje de regreso).

- **Viajes bajo demanda (*on-demand*)**. El resto de los viajes son irregulares, organizados al momento y con una planificación breve o nula. Son los más complicados de emparejar ya que requieren un sistema que funcione en tiempo real, que permita organizar parejas en el breve periodo de tiempo antes del viaje o incluso en medio del recorrido. El primer ejemplo de emparejamiento dinámico es un sistema implementado en Estados Unidos en la década de 1990, donde operadores telefónicos recibían solicitudes y formaban parejas de viaje de manera empírica.

La mayoría de los sistemas de viajes emparejados existentes funcionan sobre viajes diarios o de larga distancia, y ofrecen sistemas equivalentes a tableros de anuncios en los cuales los usuarios pueden publicar sus próximos viajes, mientras que van adquiriendo un historial y reputación en la plataforma. La implementación de sistemas de viajes compartidos dinámicos ha sido difícil, pues se requiere un sistema de *emparejamiento dinámico* que funcione en tiempo real, que pueda hallar coincidencias y ayudar a los usuarios a organizar su encuentro en un tiempo muy breve. La ventaja de un sistema de emparejamiento dinámico es que funciona para todos los tipos de viajes, y en cierto aspecto tienen mayor flexibilidad pues los usuarios no tienen que adherirse a un horario fijo. Apenas recientemente se ha vuelto posible crear sistemas automatizados de emparejamiento gracias a avances en tecnologías de comunicación, en particular se requiere un uso generalizado de dispositivos móviles con acceso a internet y GPS. Sin embargo, aunque en las últimas décadas han surgido unas pocas aplicaciones tecnológicas para viajes compartidos, ninguna ha logrado aún una adopción masiva, y la cantidad de personas viajando mediante *carpooling* continúa decreciendo, de 19.7 % de los viajes laborales en Estados Unidos en 1980, a solo 10 % en 2009 [55].

La principal dificultad que enfrentan éstos servicios es que como cualquier red social, el éxito de un sistema de emparejamiento depende de alcanzar una masa crítica de usuarios activos: si hay muy pocas ofertas la mayoría de usuarios nuevos no encontrarán viajes compartidos, y después de algunos intentos se desilusionarán y abandonarán el servicio. Pero hay simulaciones que sugieren que en ciudades grandes es posible alcanzar una población sustentable de usuarios aunque se tenga una tasa de participación relativamente baja, de unos cuantos miles de usuarios [56]. Pero es importante ofrecer incentivos a los usuarios durante un determinado periodo de lanzamiento. Una de las principales preocupaciones que mencionan los usuarios de éstos sistemas es compartir un vehículo con extraños, por lo que también es crucial implementar métodos para establecer confianza en otros usuarios – una solución común son sistemas de reputación que permitan identificar a usuarios problemáticos y construir confianza en los usuarios responsables.

2.3. La movilidad en la Ciudad de México

La Ciudad de México es una de las zonas urbanas más pobladas del mundo. Tan solo en la CDMX habitan más de 9 millones de habitantes según el Censo 2020 de INEGI, y éste número asciende a alrededor de 22 millones sí se cuenta también el área de la Zona Metropolitana del Valle de México, lo que la posiciona como la quinta ciudad más poblada del mundo, después de Tokio, Delhi, Shanghai, y São Paulo. Asímismo la Ciudad de México y el Estado de México son las dos zonas más densamente pobladas del país, con 5,967 y 724 habitantes por kilómetro cuadrado respectivamente. Tal cantidad y densidad de población está relacionada con varios de los principales problemas que enfrenta ésta urbe: contaminación ambiental, inseguridad, y una pobre movilidad urbana.

Se puede cuantificar el problema de movilidad en términos de horas perdidas al año como consecuencia de la congestión vial. Un reporte global de movilidad

de 2019 estima que en promedio los viajeros pierden 158 horas al año, clasificando a la CDMX como la tercer ciudad más congestionada del mundo, y la sexta en horas perdidas [1]. Algunos problemas empeorados por la saturación vial son la contaminación ambiental, culpable de 48 mil muertes prematuras al año en el país [3], y la inseguridad vial, siendo México el tercer país de Latinoamérica con el mayor número de muertes viales [2].

Antes de caracterizar la problemática que presenta la CDMX y ZMVM es útil discernir los factores que contribuyen a la movilidad urbana, ya que las ciudades son sistemas complejos y usualmente sus problemáticas son igualmente complejas, involucrando problemas en distintas áreas. Se han identificado al menos 8 factores interrelacionados que determinan la movilidad urbana [4]:

- **Distancia**. Cada ciudad tiene limitaciones geográficas y socio-económicas que determinan las necesidades de transporte de bienes y personas. Los viajes largos son peores que los cortos, pues ocupan recursos de transporte durante más tiempo. No habría problemas de movilidad si fuera posible que todas las personas estudiaran, trabajaran, y cultivaran comida en sus propias casas, pero difícilmente se alcanzará esa situación. Pero se pueden tomar medidas para reducir las necesidades y distancias de transporte, por ejemplo mejorando el desarrollo urbano en las áreas cercanas a las viviendas, o mediante opciones de trabajo y educación a distancia.

- **Cantidad**. Demasiados vehículos y personas saturan las vialidades y sistemas de transporte público. Para aminorar este problema, algunas ciudades han implementado medidas para desincentivar el uso de automóviles particulares, como altos impuestos y costos de estacionamientos. Los viajes en autos compartidos son otra forma de reducir la cantidad de vehículos.

- **Horarios**. Frecuentemente las principales congestiones de tráfico en ciudades ocurren en las "horas pico" (*rush hour*) de entrada y salida de oficinas y es-

cuelas. Si la gente pudiera transportarse en horarios mas flexibles se disiparía ésta congestión sobre un periodo de tiempo más grande.

- **Capacidad**. La capacidad de una vialidad o sistema de transporte determina la cantidad total de unidades que pueden transitar en ella en un tiempo dado. Se puede mejorar la movilidad urbana construyendo y ampliando autopistas, carriles de bicicletas, y sistemas de transporte público, aunque estos proyectos suelen ser costosos.

- **Infraestructura tecnológica**. La tecnología puede complementar la infraestructura existente para hacerla más óptima, aumentando su capacidad con costos reducidos. Por ejemplo, algunas ciudades han instalado sensores de tráfico en intersecciones que sirven para coordinar semáforos y reducir congestiones. Otras tecnologías útiles son los sistemas de información en tiempo real, como los GPS o las trasmisiones de radio sobre tráfico.

- **Comportamiento**. Un comportamiento inadecuado de los transeúntes ocasiona ineficiencias y demoras. Suele ocurrir cuando los viajeros siguen estrategias de tránsito egoístas o inatentas, como el uso de celular mientras se conduce, cambios compulsivos de carril, y no respetar señales de tránsito o semáforos. Mejorar éste aspecto requiere intervenciones para promover comportamientos positivos y restringir actitudes negativas, por ejemplo con campañas de educación y multas de tránsito.

- **Sociedad**. Existen varios aspectos sociales que influyen en otros factores de movilidad. En la mayoría de las sociedades tener un automóvil particular es un símbolo de estatus y éxito económico, incentivando su uso. Para disminuir el uso de automóvil es necesario seguir aumentando la aceptación social de otros medios de transporte como bicicletas y transporte colectivo, además claro de proporcionar la infraestructura adecuada.

- **Planeación y regulación**. Las ciudades sufren ineficiencias cuando hay poca o nula planeación urbana. En muchas ciudades se complica éste aspecto ya que las decisiones de proyectos urbanos no recaen en urbanistas sino en políticos quienes usualmente siguen motivaciones demagógicas, y no siempre buscan la intervención más eficiente o que beneficiaría a más personas. Asímismo se requiere coordinar una regulación o legislación efectiva para dar seguimiento a los proyectos planeados, ya que en algunas ciudades se planifica adecuadamente pero las propuestas no se materializan porque ningún organismo se encarga de que se apliquen los cambios.

La situación de la CDMX tiene problemáticas que afectan varios de éstos factores. En primer lugar, históricamente su desarrollo urbano ha sido altamente centralizado. Desde épocas prehispánicas el área del centro histórico ha albergado los principales asentamientos de la región, así como el gobierno local y nacional, y hasta inicios del siglo XX la mayoría del crecimiento urbano se concentró en la zona de 20 kilómetros alrededor del Zócalo, la plaza principal. Desde entonces la distribución poblacional se fue separando entre zonas residenciales de élite cercanas al centro, y viviendas económicas de gente de bajos recursos usualmente alejadas o ubicadas en la periferia [57]. El crecimiento poblacional de ésta urbe continuó a lo largo de todo el siglo pasado bajo ésta dinámica de separación socio-económica, hubo un enorme desarrollo de residencias baratas en distritos periféricos cada vez más alejados, eventualmente llegando a los estados contiguos como el Estado de México y Hidalgo. Sin embargo la descentralización de viviendas no se vio reflejada en los servicios de salud, educativos, financieros y laborales, los cuales permanecieron concentrados en su mayoría en áreas centrales de la Ciudad de México. Hoy en día es muy notoria ésta concentración de servicios, y estudios han hallado que cuatro distritos del centro de la ciudad son destino de aproximadamente 50 % de los viajes, generan el 53 % del total de empleos formales y los sueldos promedio por persona superan cinco veces

las demarcaciones en la periferia [57].

Por éstos motivos una gran cantidad de personas viven a una distancia considerable de su trabajo o escuela, y aceptan los costos y tiempos de traslado adicionales por motivos económicos. Según la La Encuesta Origen - Destino 2017 por INEGI [8] al menos 19 % de los habitantes de la CDMX (1 720 145 personas) son población "flotante" que habitan en otros estados, principalmente el Estado de México, Hidalgo y Morelos, y se trasladan diariamente para estudiar o trabajar. De ellos entre 39 % y 47 % tardan de una a dos horas en su traslado. Asímismo estima que en promedio la gente tarda 1 hora en llegar al trabajo, 30 minutos para ir a escuelas, y 45 minutos en regresar al hogar, y otras encuestas estiman tiempos aún mayores [58].

Pero el problema no es solo la cantidad de viajes y las grandes distancias recorridas diariamente, sino los horarios en que se aglomeran. Como el motivo de muchos de éstos viajes es ir al trabajo o escuela, una enorme cantidad de viajes se concentran alrededor de los horarios laborales y escolares, causando congestiones diarias en las horas de entrada y salida de la ciudad (figura 2.6). La mayor congestión es la matutina, y mas de 4 millones de viajes se inician diariamente entre 7:00 y 7:59 A.M.

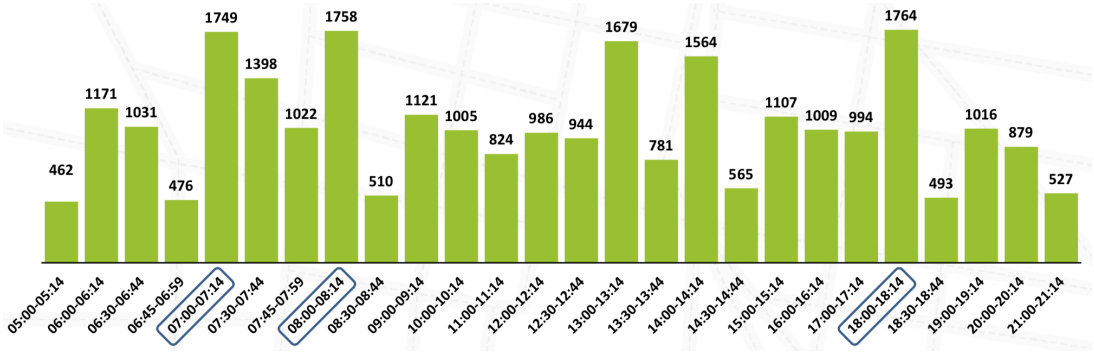

Figura 2.6: Número de viajes (en miles) realizados un día entre semana según la hora de inicio del viaje. Fuente: INEGI Encuesta Origen - Destino en Hogares de la Zona Metropolitana del Valle de México (EOD) 2017.

De todos los medios de transporte que se ocupan en la CDMX y ZMVM, los automóviles son los que más contribuyen a la congestión vial, pues ocupan mucho más espacio por cada pasajero transportado que cualquier otro método (figura 2.7). La mayoría de los viajes en automóvil transporta a solo uno o dos pasajeros: en la ZMVM de los 6.60 millones de viajes en automóvil en el 90.5 % de ellos el conductor inicia viajando a lo mas con otra persona, y el promedio de ocupantes por carro es de 1.5 personas. Y los autos también consumen una cantidad mucho mayor de espacio en estacionamientos, por ejemplo en todo México entre 2009 y 2013 de los más de 16 millones de metros cuadrados de desarrollo inmobiliario reportados, 42 % son estacionamientos [59]. Así que no es sorpresivo que los automóviles acaparen la mayor parte del espacio vial de la CDMX a pesar de que son usados por un porcentaje relativamente pequeño de la población – se estima que el auto es parte de tan solo entre 22.3 % y 29 % de todos los viajes. Además de caminar que es parte de 65 % de todos los viajes, el transporte público es el método más popular y forma parte de entre 50 % y 60 % de todos viajes, de los cuales el tipo más usado son los colectivos (ie. microbuses, combis), usados en aproximadamente 75 % de los viajes en trasporte público, seguidos del Metro con 30 %. Las bicicletas solo figuran en 2.2 % de todos los viajes [8].

A la larga una solución deseable al problema de movilidad es aumentar el desarrollo urbano en los distritos y estados periféricos, para ofrecer allí mayores oportunidades laborales y escolares. Naturalmente también se puede mejorar la movilidad incrementando la infraestructura de transporte público y vialidades. Pero reducir la cantidad de automóviles es también una posibilidad prometedora, y la aplicación tecnológica descrita en ésta tesis está orientada hacia éste fin. Es el objetivo principal de un sistema de viajes compartidos, y cada emparejamiento exitoso implica dos optimizaciones, aprovecha los asientos vacíos que de cualquier manera iban a consumir espacio y gasolina, y descongestiona los sistemas de transporte ya que o reduce la cantidad de autos o la cantidad de usuarios de transporte público. Además

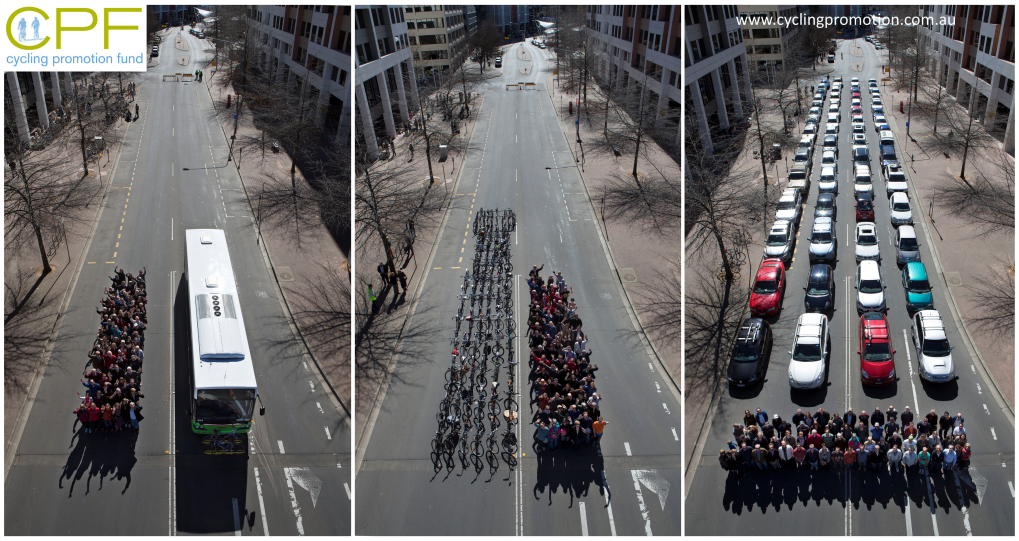

Figura 2.7: Comparación visual del espacio requerido para transportar 70 personas en autobús, bicicleta, y automóvil, de izquierda a derecha. Fuente: Cycling Promotion Fund Australia, 2011.

la situación de movilidad de la CDMX implica que hay muchos viajes posiblemente emparejables, ya que una gran cantidad de viajes coinciden en dirección y horario, van de la periferia hacia el centro y viceversa alrededor de los mismos horarios laborales.

Capítulo 3

PumaMóvil: Descripción de componentes

Éste capítulo contiene una descripción general de los sistemas componentes de la aplicación. Posteriormente el capítulo 4 detalla los sistemas implementados por el autor, incluyendo capturas de pantalla de las interfaces de usuario en iOS y Android, una especificación de servicios REST, y diagramas UML de los modelos agregados a la base de datos.

El objetivo central de ésta aplicación es promover la adopción de nuestro sistema de viajes compartidos dinámicos (en tiempo real, según la demanda) en la población de miembros activos de la UNAM. Adicionalmente, se han incluido en la aplicación diversas funcionalidades secundarias para que resulte más atractiva para usuarios potenciales. Con el fin de incrementar la percepción de seguridad de los usuarios, durante el registro de usuarios se valida su afiliación a la UNAM comparando su número de cuenta o de trabajador con bases de datos de DGTIC. Para atraer y retener usuarios recurrentes, se han anexado a la aplicación utilidades informativas relativas a movilidad movilidad, que permiten conocer la ubicación de Pumabuses en tiempo real y el inventario de módulos de Bicipuma. Asimismo para mayor seguridad de los

usuarios se implementó un 'Botón S.O.S.', el cual les permite contactar rápidamente a la policía local o Vigilancia UNAM, y compartir la ubicación del GPS en tiempo real con un contacto de su elección.

3.1. Infraestructura de software

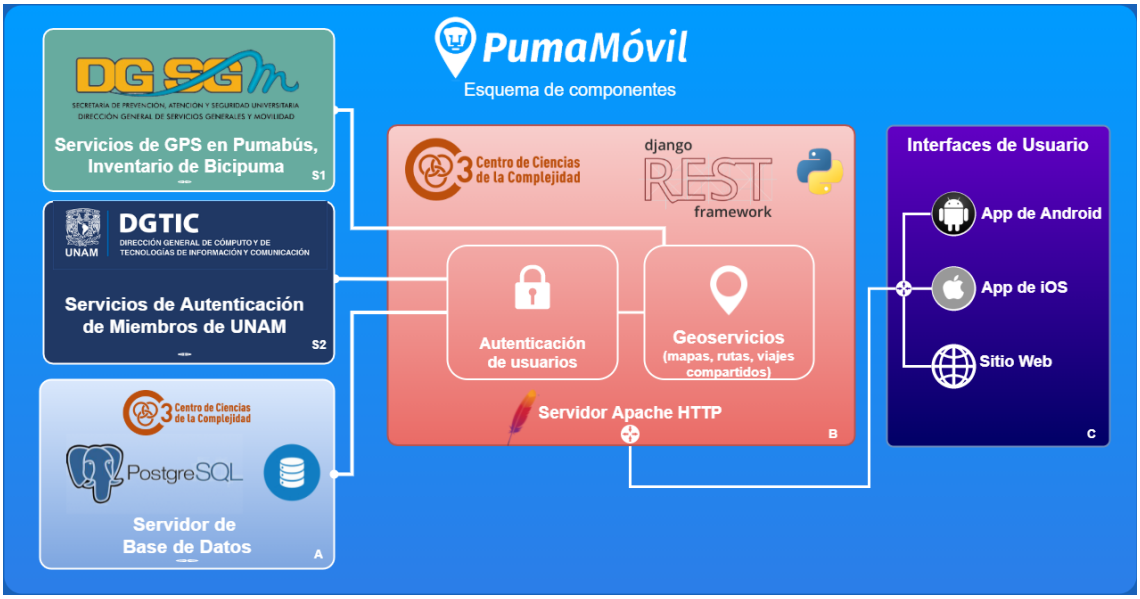

Figura 3.1: Diagrama de sistemas componentes de la aplicación. Imagen elaborada por el autor.

Para otorgar funcionalidad a los diversos sistemas que ofrecen las interfaces de usuario, se requirió la integración de conexiones entre diversos proveedores de servicios. El esquema de la figura 3.1 ilustra los canales de comunicación entre los principales componentes de la aplicación. De derecha a izquierda, las interfaces de usuario o 'Front-End' (recuadro de color morado de la figura) se conectan a un único servidor ubicado en el C3 (recuadro de color rojo), el cual funge como 'nexo' entre todos los sistemas componentes, y recibe todas las peticiones de servicios geográfi-

cos (ruta más corta, ubicación de recursos, viajes compartidos) y de autenticación de usuarios (registro, login, editar perfil). Dependiendo del servicio solicitado, éste servidor central se comunica con: el servidor que aloja la base de datos, ubicado asimismo en el C3 (recuadro azul claro), el servicio externo que provee DGTIC para validar miembros de UNAM (recuadro azul oscuro), y los servicios de la DGSGM que ofrecen la información más reciente de ubicaciones de Pumabuses y inventarios de módulos de Bicipuma (recuadro verde).

Una consideración importante del planteamiento de este proyecto fue que todo el código sea desarrollado usando software gratuito y de uso libre (*open source*), para evitar atar el futuro del producto a herramientas que requieran pagos periódicos o que pueden no estar disponibles en el futuro. Ambos servidores alojados en el C3 cuentan con Linux (Ubuntu) como sistema operativo. El Sistema Manejador de Base de Datos (SMBD) es PostgreSQL, una extensión de SQL que ofrece herramientas para el manejo de información geográfica y de mapas, además de manejar eficientemente grandes cantidades de datos. El servidor de aplicación aloja tanto el sitio web del proyecto, como los servicios REST consumidos por las aplicaciones de Android e iOS. Éstos servicios REST están programados en Django Framework bajo el lenguaje Python 3.5. El software que aloja el proyecto de Django y maneja las conexiones entrantes y salientes es Apache HTTP Server. Las dos aplicaciones nativas en Android e iOS están programadas respectivamente en Android Studio con Java, y Xcode con el lenguaje Swift.

Nuestro servidor de base de datos se encuentra separado del servidor web o de aplicación, ya que ésto ofrece varias ventajas [60, 61]. La base de datos adquiere una capa adicional de seguridad pues ésta separación permite limitar las aplicaciones instaladas y puertos abiertos en el servidor de base de datos a solamente los que sean estrictamente necesarios. El servidor de base de datos es más estable pues se encuentra más aislado de posibles fallas críticas del servidor de aplicación, el cual usualmente requiere instalar más dependencias y recibe conexiones de diversas apli-

caciones, por lo que tiene más posibles puntos de fallo o ataque. Adicionalmente, ésta separación facilita identificar el origen de errores, evaluar el desempeño de cada componente por separado, así como permite adecuar el balance de carga y características del hardware dependiendo del tipo de procesos que realiza cada servidor.

3.2. Registro y autenticación de miembros de UNAM

Como se mencionó previamente, ésta aplicación contempla tres tipos de usuarios: estudiantes activos de la UNAM, trabajadores de cualquier área de la UNAM, y público general. Las primeras dos categorías tienen acceso a todas las funcionalidades ofrecidas en la app, mientras que el público general tiene acceso a todo excepto al sistema de creación viajes compartidos.

Para validar el estatus de estudiantes y personal activo de UNAM, la Dirección General de Cómputo y Tecnologías de Información y Comunicación (DGTIC) de la UNAM nos facilitó el acceso a dos servicios de validación. Estos servicios no nos proporcionan ninguna información privada del usuario, únicamente responden una bandera indicando si los datos proporcionados corresponden a un estudiante/trabajador activo o no.

Nuestro sistema de registro de usuarios recibe los datos necesarios según el tipo de usuario, los cuales se comunican al servicio de DGTIC correspondiente para verificar su validez, una vez obtenida una respuesta se crea el usuario o se devuelve un mensaje de error según sea el caso. El diagrama de la figura 3.1 ilustra las conexiones mencionadas, y las interfaces de registro para los usuarios están detalladas e ilustradas en la sección 4.1.1.

Nombre	Validar estudiante UNAM.
Acción	Una interfaz desea consultar si los datos proporcionados corresponden a un estudiante activo de la UNAM.
Parámetros de entrada	▪ Número de Cuenta (String) ▪ Nivel (String) ▪ Plantel (Int)
Respuesta	▪ $HTTP\_200\_OK$ si los datos corresponden a un estudiante activo (que al menos ha inscrito una materia éste semestre). ▪ $HTTP\_400\_BAD\_REQUEST$ si no hay registro o el estudiante no es activo.
Descripción	El servicio valida que el número de cuenta exista y que corresponda a un estudiante inscrito en el nivel y plantel especificado. El parámetro "nivel" es un ID que indica si el estudiante está en Bachillerato, Licenciatura, Maestría, etc. El plantel es un ID numérico entre 1 y 713. Éstos códigos fueron especificados por DGTIC.

Servicio de DGTIC para validación de estudiantes UNAM.

Nombre	Validar trabajador UNAM.
Acción	Una interfaz desea consultar si los datos proporcionados corresponden a un trabajador activo de la UNAM.
Parámetros de entrada	▪ RFC (String)
Respuesta	▪ $HTTP\_200\_OK$ si el RFC corresponde a un trabajador activo de la UNAM. ▪ $HTTP\_400\_BAD\_REQUEST$ si no hay registro con ese RFC o el trabajador no es actualmente activo.
Descripción	El servicio solamente valida que el RFC exista en la planta activa de trabajadores de alguna dependencia de la UNAM, y devuelve la bandera correspondiente.

Servicio de DGTIC para validación de trabajadores UNAM.

Una limitación de éstos servicios de validación es que solamente responden afirmativamente si son actualmente miembros activos, los ex-alumnos no se pueden validar ya que no existe ninguna base de datos centralizada que podamos consultar. Para ellos y todos los demás usuarios potenciales se ofrece el registro general abierto a todo el público que solamente requiere un correo electrónico, contraseña, y nombre completo.

3.3. Asignación de viajes compartidos

El proceso para que los usuarios busquen y organicen opciones de viajes compartidos involucra dos problemas de optimización, el de hallar la ruta más corta entre

dos puntos, y la búsqueda de emparejamientos viables entre las rutas existentes. El flujo de pantallas de ambos procesos está ilustrado y explicado en las secciones 4.1.4 y 4.1.5, ésta sección solo detalla el funcionamiento general de los algoritmos usados.

3.3.1. Algoritmo de ruta más corta

La formulación de éste problema es la siguiente: se tiene una red (grafo) conformada por dos conjuntos, uno de vértices y uno de aristas entre vértices, donde cada arista tiene un peso $w \geq 0$. El problema de optimización consiste en hallar la secuencia de vértices contiguos (existe una arista entre cada par de vértices subsecuentes), donde la suma de los pesos de todas las aristas en la secuencia es la mínima posible.

Se han propuesto una gran variedad de soluciones, la más conocida es el algoritmo de Djikstra pues resuelve en su mayoría éste problema. En resumen este realiza una búsqueda por amplitud (Breadth First Search) a partir del nodo de origen, y en cada nodo recorrido va anotando la secuencia de menor peso hasta el origen. Eventualmente se va a haber recorrido toda la red que sea alcanzable desde el origen, y el nodo destino tendrá anotada la secuencia hacia el origen de menor peso. Su desventaja principal es que requiere mucha memoria de trabajo cuando el número de nodos es muy grande. Han surgido diversas optimizaciones sobre éste algoritmo, entre ellas Djikstra bidireccional que busca simultáneamente alrededor de ambos nodos de origen y destino, el cual es más rápido en muchos casos [62].

Nuestro sistema de trazado de rutas ocupa la implementación de PostgreSQL del algoritmo de Djikstra bidireccional. Otros algoritmos de búsqueda notables incluyen el ‘reach based routing’ y la ‘búsqueda A*’, los cuales usan heurísticas para complementar la búsqueda basadas en otros valores, como la conectividad de cada nodo y la distancia euclidiana hasta el nodo destino.

3.3.2. Algoritmo de emparejamiento de viajes

Una vez que se calculó una ruta $X = N_1, ..., N_k$ definida simplemente como una secuencia de nodos, el problema de emparejar rutas similares involucra varias casos posibles. El caso más sencillo es que una de las dos rutas se encuentre contenida completamente en la otra, y la posibilidad de compartir esos viajes es obvia. Pero el otro caso es que sean rutas en las cuales ambas tienen un segmento 'paralelo' hacia la misma dirección, pero viajar juntos requeriría necesariamente desvíos de uno o ambos participantes. Para estos casos el algoritmo debe considerar una cota superior de la distancia máxima de desvío, así como proponer un punto de encuentro, despido, y ruta compartida nueva.

Nuestro sistema de emparejamiento implementa un algoritmo propuesto por Schreieck et al. en 2016 [63], el cual se enfoca en alcanzar breves tiempos de respuesta para poder procesar grandes cantidades de ofertas y demandas en tiempo real. Su optimización crucial consiste en una estructura de datos de índices invertidos, que básicamente es una tabla donde cada nodo tiene asociado una lista de los IDs (identificadores únicos) de todas de las rutas que pasan por ahí. Cuando se desea buscar posibles emparejamientos de una ruta nueva, se usa la estructura de índices invertidos para hallar rápidamente todas las rutas que pasan por los nodos que se encuentran a un cierto radio del punto de origen y destino. Éste radio corresponde a la distancia máxima de desvío que se considera aceptable, y puede ser preestablecida o definida por el usuario. Éste algoritmo se puede extender para incorporar otras limitaciones, por ejemplo en el caso de viajes en automóvil es necesario considerar que el número de asientos que ofrece el conductor sea mayor o igual a los que necesita el pasajero.

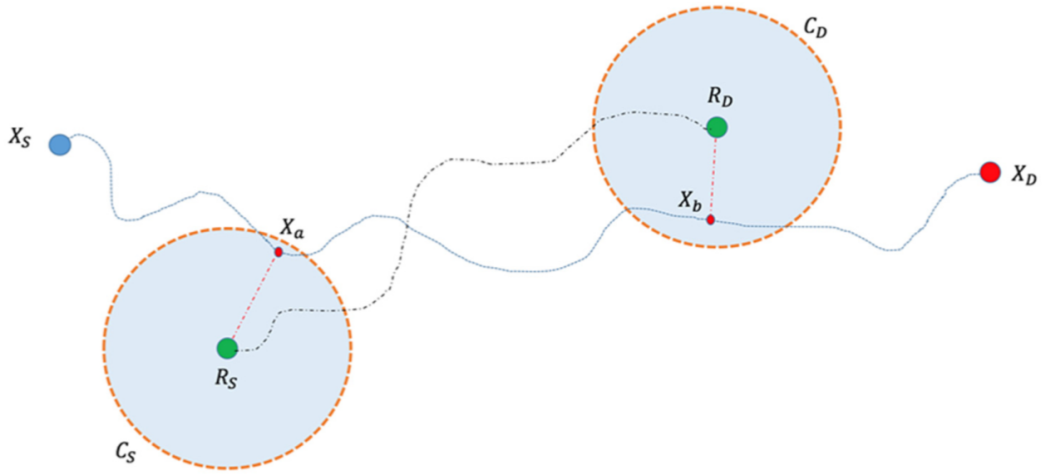

Figura 3.2: Ilustración de un emparejamiento entre dos rutas. La ruta azul entre X_s y X_d representa una oferta existente, cuando se crea la ruta verde entre R_s y R_d se realiza una búsqueda alrededor del nodo origen y destino, hallando en éste caso los nodos X_a y X_b como posibles puntos de encuentro y despido. Fuente: Schreieck et al. 2016.

3.4. Recursos informativos en tiempo real

El principal propósito de ofrecer recursos informativos de movilidad es atraer y retener usuarios de la aplicación. Pero como se mostró en la sección 2.2.4 del marco teórico, el simple hecho de ofrecer información relevante es por sí mismo una solución de movilidad, pues facilita la auto-organización de los usuarios al reducir ineficiencias en su transporte.

Como la población objetivo de usuarios son miembros de la UNAM en la ZMVM, la mayoría de ellos son visitantes de Ciudad Universitaria y les resulta relevante información sobre las rutas de Pumabuses más cercanas, la cantidad de bicicletas disponibles en cada módulo de Bicipuma, y la ubicación de diversos elementos de seguridad como casetas de vigilancia, postes de auxilio, rampas de accesibilidad, etc. Asímismo, se ofrece la opción de consultar en el mapa las líneas y estaciones del

Metro y Metrobus de la CDMX. Las interfaces para visualizar éstos recursos están explicadas e ilustradas en la sección 4.1.6 y 4.2.3.

Éstos recursos informativos provienen de dos fuentes. Los recursos 'estáticos' son aquellos que cambian poco frecuentemente – por ejemplo las rutas de Pumabús, Metro, Metrobus, postes de seguridad y rampas – y se encuentran alojados en nuestro servidor de base de datos, las interfaces de usuario los solicitan a través de servicios del servidor de aplicación. La ventaja de alojarlos en nuestro servidor, en lugar de directamente en las aplicaciones móviles, es que si hay cambios en ellos se pueden actualizar las tablas de datos, y este cambio se ve reflejado inmediatamente en todas las interfaces, de otra manera se tendría que publicar una nueva versión en tiendas lo cual es más tardado.

Por otro lado están los recursos "dinámicos" cuyo estado cambia en tiempo real en cuestión de segundos, éstos incluyen las ubicaciones actuales de Pumabuses y los inventarios de módulos de Bicipuma. Éstos datos provienen de servicios alojados por la Dirección General de Servicios Generales y Movilidad (DGSGM) de la UNAM. Nuestro servidor de aplicación se comunica con la DGSGM para obtener y devolver a los usuarios la información más reciente, éste esquema de comunicación se ilustra en la figura 3.1. El servidor de DGSGM cuenta con un sistema de monitoreo de Pumabuses que recibe las ubicaciones más recientes de GPS instalados en cada vehículo de Pumabus. Asímismo cada módulo de Bicipuma cuenta con un sistema web que permite actualizar el conteo de bicicletas que se tienen en inventario cada que se reciban o presten. Éstos conteos se comunican al servidor central de DGTIC, el cual a su vez ofrece un servicio para obtener los valores más recientes en formato JSON.

3.5. Herramientas de seguridad

Hemos incluido funcionalidades para mejorar la seguridad personal de los usuarios, pues su seguridad es una de las principales preocupaciones que plantean los usuarios sobre un sistema de viajes compartidos. Por este motivo solo se pueden crear viajes compartidos entre miembros de la UNAM [9].

La aplicación cuenta con mapas de recursos informativos de seguridad, así como un botón de emergencia 'S.O.S.' en la pantalla principal que permite rápidamente llamar a la policía de la CDMX o a Vigilancia UNAM, dependiendo de la ubicación del usuario. Asímismo, hay otro botón para compartir ubicación en tiempo real, el cual generar inmediatamente un enlace web que el usuario puede compartir con cualesquiera contactos de su elección para permitirles seguir su posición en tiempo real, esto sin necesidad de acceder a la aplicación o tener una cuenta de usuario.

Capítulo 4

Contribuciones al desarrollo del proyecto

4.1. Aplicación de Android

En la aplicación para dispositivos Android mi labor consistió en implementar en la app el diseño de pantallas que fue aprobado por los responsables del proyecto, y conectar la interfaz de usuario con el servidor en diversos puntos mediante llamadas HTTP. Cada sección incluye capturas de las pantallas desarrolladas, generadas en un simulador de un celular Pixel 2 XL con Android 10.0.

4.1.1. Registro de usuarios

La app cuenta con tres formularios de registro para los usuarios nuevos, para estudiantes de UNAM, trabajadores UNAM, y público general. La interfaz consiste en una sola actividad o pantalla de registro, con una barra superior para alternar entre las tres opciones. Al abrir ésta pantalla por primera vez se muestra un breve diálogo de texto explicando los tres tipos de registro.

El registro de público general solo requiere nombre, correo y contraseña. A los

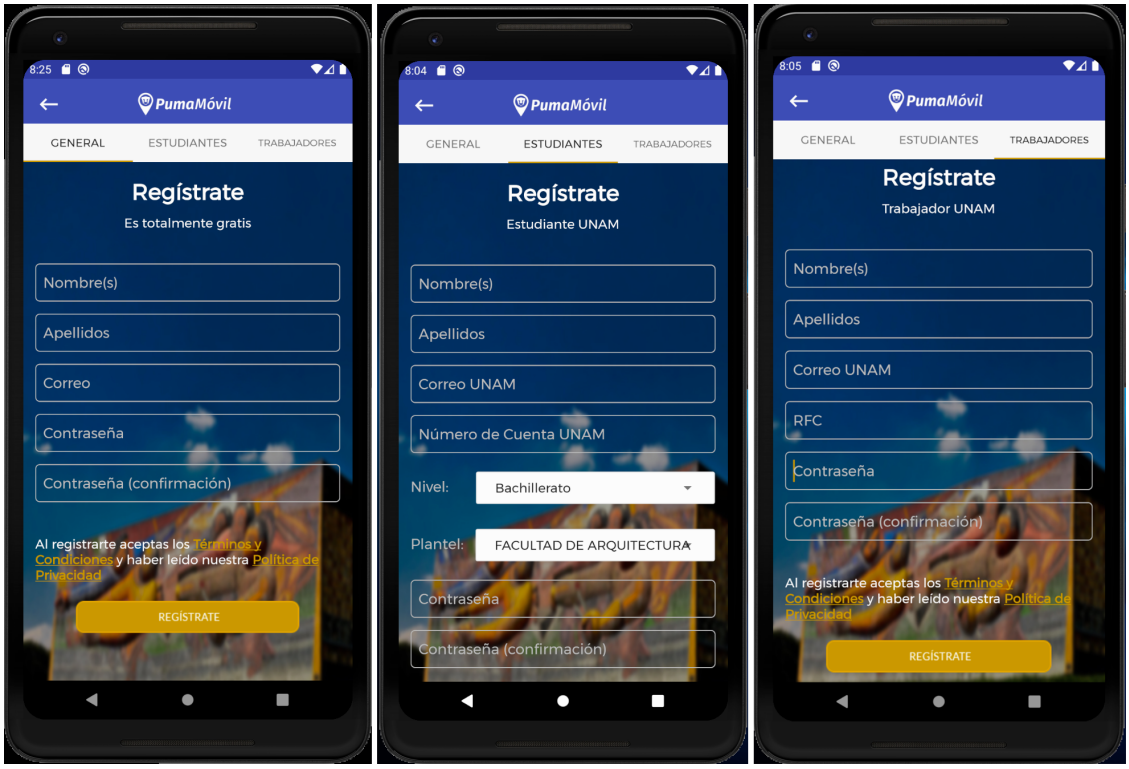

Figura 4.1: Los formularios de registro de usuarios. Imagen elaborada por el autor.

trabajadores de UNAM se les solicita adicionalmente su RFC. A los estudiantes de UNAM se les solicita adicionalmente su número de cuenta, nivel (Bachillerato, Licenciatura, etc.), y plantel. Estos últimos dos datos se seleccionan de una lista desplegable de opciones predefinidas.

De ser exitosa su validación como miembros de UNAM, nuestro servidor valida que no exista una cuenta asociada a los datos proporcionados, de ser así se crea la nueva cuenta de usuario pero se marca como inactiva. Para concluir el proceso de registro y activar la cuenta solamente se requiere que el usuario acceda a una URL que le fue enviada a su correo electrónico, para validar el mismo.

4.1.2. Inicio de sesión

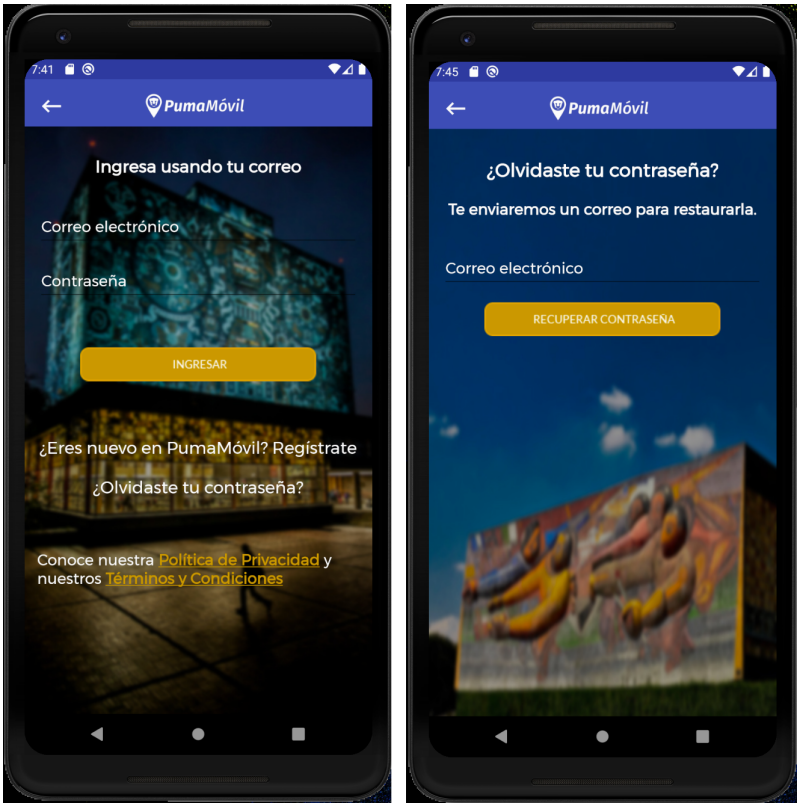

Figura 4.2: Las pantallas de inicio de sesión y recuperar contraseña. Imagen elaborada por el autor.

El inicio de sesión se realiza mediante correo electrónico y contraseña. La primer vez que se inicia sesión en un dispositivo se pide que el usuario confirme que acepta los Términos y Condiciones y la Política de Privacidad. La pantalla de inicio contiene enlaces a los Términos y Condiciones, la Política de Privacidad, y al servicio para recuperar una contraseña olvidada.

Una vez que el servidor ha validado los datos de acceso éste genera un Token de sesión de usuario, el cual se almacena en la base de datos y se comunica al dispositivo usuario en la respuesta a su petición HTTP. Éste token se requiere como parámetro

para consumir todos los servicios (excepto el registro y login), y en cada conexión el servidor compara el token recibido con los que ha guardado en su base de datos, lo que valida que la petición provenga de un usuario que ha iniciado sesión.

4.1.3. Rediseño de pantalla principal

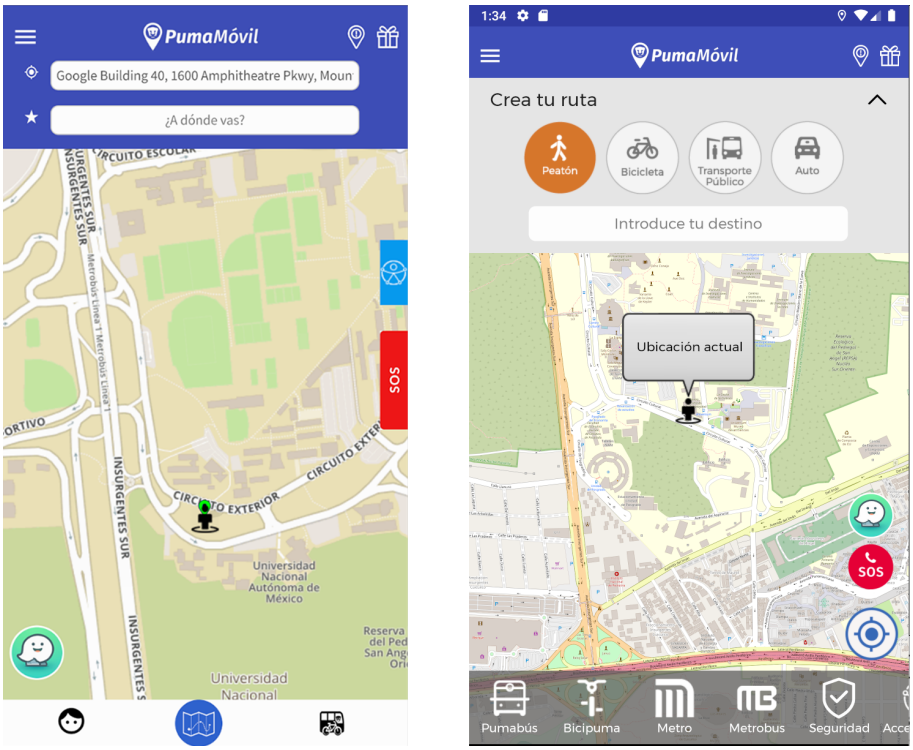

Figura 4.3: Las pantallas principales de las versiones 1.0 y 2.0 de la app, respectivamente. Imagen elaborada por el autor.

La versión de la app PumaMóvil que se describe en éste trabajo es la 2.0. La principal novedad en ésta versión es todo el sistema para organizar viajes compartidos en bicicleta, pero su lanzamiento se ha postergado hasta que se reanuden las clases presenciales en UNAM. Sin embargo, actualmente ya se encuentra disponible en las tiendas Play Store y App Store la versión 1.0 de ésta app. Ésta primer ver-

sión cuenta con todas las funcionalidades de la 2.0 exceptuando las siguientes: no existe el sistema de búsqueda de viajes compartidos entre bicicletas, no hay manera de visualizar rutas de Metro o Metrobús, y solamente se puede buscar la ruta más corta en transporte público.

Para poder incorporar el nuevo sistema de viajes compartidos, así como para atender observaciones de usabilidad hechas por usuarios de la primera versión, en la segunda versión se rediseñó casi totalmente la pantalla principal de la app. La pantalla principal o *home screen* de una aplicación es muy importante pues debe servir como nexo para acceder a las distintas funcionalidades que ofrece, por ello es deseable optimizar su usabilidad aplicando varios principios de diseño de aplicaciones móviles [64, 65, 66]. Los principales detalles identificados en su primer versión son los siguientes:

- La pantalla para consultar la ubicación actual de los Pumabuses cercanos es la función más usada de ésta versión de la app, sin embargo no es inmediatamente claro cómo acceder a ella. Se abre mediante el botón derecho de la barra inferior de navegación. Este botón permite acceder a los recursos de Pumabús y de Bicipuma, por ello su icono es un Pumabus y una bicicleta, pero su significado no es claro para todos los usuarios, y en pantallas de baja resolución no se distingue bien el dibujo.

- No es inmediatamente claro para qué sirven los botones laterales (del lado derecho de la pantalla). Éstos permiten acceder a las funciones relativas a Accesibilidad (llamar al transporte en C.U., ver rampas en mapa) y Seguridad (llamar a la policía o seguridad UNAM, compartir ubicación, ver recursos de seguridad en mapa).

- El diseño rectangular de los botones laterales los hace difícil de presionar, en particular en dispositivos con menor tamaño de pantalla o usuarios con dedos grandes.

- La pantalla de perfil de usuario, accesible mediante el botón izquierdo de la barra inferior de navegación, tiene la misma prioridad visual que las otras dos de esa barra, pero no tiene funcionalidad más que mostrar el nombre y correo electrónico.

La versión 2.0 de la aplicación introduce los siguientes cambios:

- Se eliminó la barra inferior de navegación y en su lugar se creó una nueva barra inferior semitransparente, la cual contiene todos los botones que permiten visualizar recursos informativos en el mapa. Éste acomodo tiene varias ventajas:
  - Es más claro para qué sirve cada botón gracias a sus respectivas etiquetas de texto.
  - En cuanto a usabilidad es más fácil encontrar un recurso informativo deseado, ya que en lugar de agrupar el contenido de la app según su tema (los mapas de seguridad en el botón S.O.S., los mapas de rampas en el botón de Accesibilidad, etc.), se agrupan según su funcionalidad (todos los botones que dibujan recursos en el mapa están juntos e inmediatamente accesibles).
  - Le da mayor visibilidad a recursos que se encontraban dispersos en pantallas secundarias.
  - Optimiza la velocidad de la app, pues todos estos recursos se dibujan en el mismo mapa en lugar de tener que instanciar una nueva pantalla o Actividad para cada uno.
- La barra para introducir Origen/Destino conserva la mayor prioridad visual en la parte superior de la pantalla, pero fue rediseñada para incorporar las nuevas funcionalidades de rutas:

  - Se agregaron 4 botones para seleccionar un tipo de transporte. Para hacer espacio a estos botones se eliminó el cuadro de texto para seleccionar origen, el cual por default es la ubicación actual del usuario pero se puede modificar durante el flujo para confirmar viaje (detallado en la sección 4.1.4).
  - En lugar de mostrar la ruta más corta hallada directamente en la pantalla principal, el botón de "Introduce tu destino" inicia un flujo (detallado en la sección 4.1.4) que contiene pantallas que permiten editar el origen y destino, así como pantallas para guardar y acceder a direcciones favoritas y del historial de destinos.

- Como se movieron funciones de los botones laterales a la nueva barra inferior, los botones laterales solo contienen las funciones para llamar al transporte en C.U., llamar a la policía o vigilancia UNAM, y compartir ubicación. Éstos botones se unificaron en un solo botón S.O.S., el cual ahora es redondo para que sea más fácil de tocar, y muestra un icono de teléfono para mayor claridad de contenido.

- Se agregó un botón que centra el mapa en la ubicación actual del usuario. Este botón se agrupó con los botones de S.O.S. y Waze Carpool en la esquina inferior derecha de la pantalla, para reducir el desorden visual y liberar el espacio en la esquina inferior izquierda para mostrar las listas desplegables de rutas de Pumabús y recursos de seguridad.

4.1.4. Creación de rutas

Antes de poder organizar viajes emparejados, es necesario un flujo en el cual los usuarios especifiquen los parámetros de sus próximos viajes. Esencialmente necesitamos capturar el origen, destino y tipo de transporte del viaje. Con éstos datos podemos calcular y mostrar al usuario la ruta más corta según el medio de transporte solicitado, y buscar otros viajes similares si así lo desea.

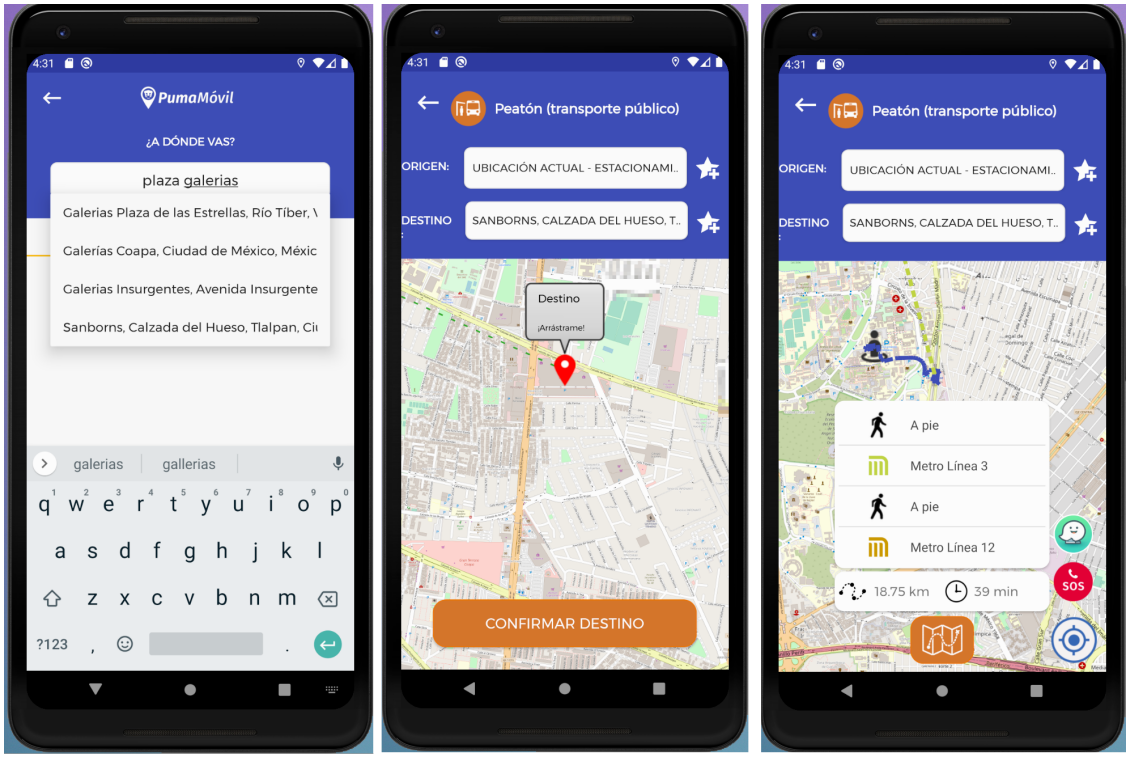

Figura 4.4: Pantallas de: (1) búsqueda de direcciones cercanas, (2) confirmación de destino, y (3) desglose de la ruta más corta hallada. Imagen elaborada por el autor.

El punto de inicio de éste flujo se encuentra en la parte superior de la pantalla principal (figura 4.3). Lo primero que se requiere es seleccionar una de las cuatro opciones de transporte disponibles: automóvil, bicicleta, a pie, y transporte público. Después el usuario puede introducir la dirección de su destino en el cuadro de texto

que se encuentra abajo (figura 4.4), éste mostrará un listado de ubicaciones que coinciden según su cercanía a la ubicación actual del usuario. Una vez que el usuario selecciona una opción, la siguiente pantalla sirve para confirmar el destino, muestra un marcador de ubicación del destino en un mapa para que el usuario confirme que es correcta, o arrastre el marcador a la posición correcta. Hasta éste punto se asume que el origen del viaje es la ubicación actual del dispositivo, pero en caso contrario la pantalla para confirmar destino incluye un cuadro de búsqueda para introducir el origen deseado. Una vez que se presione el botón de 'Confirmar viaje' se envían los datos al servidor para calcular la ruta más corta, misma que se dibuja en el mapa, junto con un desglose en texto de los pasos del viaje, la distancia total y el tiempo estimado.

La búsqueda de direcciones realiza consultas a una API de libre acceso de OpenStreetMaps llamada Nominatim. Para usar el servicio de autocompletar dirección, el servicio recibe el texto a empatar y devuelve un listado en formato JSON de coincidencias con las coordenadas, nombre, dirección y otros detalles de cada ubicación. Los nombres se usan para rellenar dinámicamente el menú desplegable de opciones, y cuando el usuario selecciona una opción se usa el índice numérico de la celda seleccionada para obtener de un arreglo las coordenadas correspondientes.

Se necesitó almacenar los datos del viaje actual de una forma semi-persistente, ya que el flujo transcurre en varias pantallas. Para esto se creó un objeto global de acceso público llamado 'UsuarioPumaMovil', el cual se instancia al iniciar sesión y almacena entre otros datos la ubicación actual, el token de conexión, el origen, destino, y tipo de transporte del viaje actual. Esta información no necesariamente debe perdurar si se cierra la app, así que no fue necesario guardarla en la memoria de almacenamiento interno.

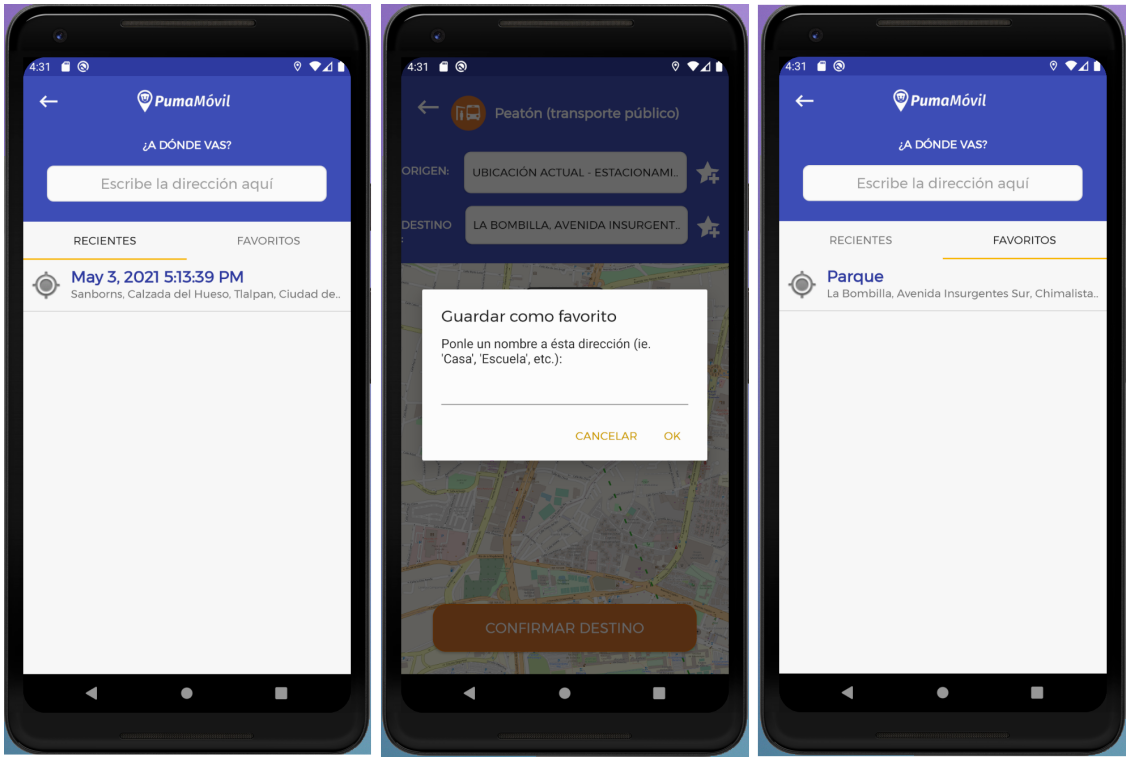

Figura 4.5: (1) Listado de destinos recientes, (2) diálogo para agregar una nueva ubicación favorita, y (3) listado de destinos favoritos. Imagen elaborada por el autor.

Almacenamiento de direcciones favoritas y recientes

Para ofrecer una comodidad de uso a los usuarios frecuentes, además del cuadro de búsqueda de direcciones los usuarios pueden seleccionar una dirección anteriormente visitada, ya sea de la lista de direcciones favoritas que agregan manualmente o de un listado con el historial de todos los destinos consultados. Cuando el usuario presiona el campo de texto para buscar una dirección, en realidad se está presionando un botón que abre una nueva pantalla, la cual además del cuadro de búsqueda contiene dos pestañas listando las direcciones recientes y favoritas (figura 4.5). Las direcciones recientes se guardan cada vez que el usuario empieza un nuevo viaje, y las favoritas se agregan manualmente con el botón de estrella que aparece al lado

de cada campo de texto. Los destinos recientes se muestran con la fecha y hora en que se realizó el viaje, y las favoritas con el nombre que le puso el usuario.

Para almacenar tanto las direcciones recientes como favoritas se ocupó un objeto ArrayList cuyos elementos son objetos de tipo 'Direccion', que contiene los atributos de latitud, longitud, nombre, y calle. Para que persistan los datos aunque se cierre la app, éstas listas se deben almacenar en la memoria interna del dispositivo. Para ésto las listas se codifican como cadenas de texto de tipo JSON, las cuales se almacenan en las SharedPreferences de Android. Así, el proceso para guardar una dirección nueva es obtener la cadena en formato JSON de SharedPreferences, decodificarla en un nuevo ArrayList, agregar la nueva dirección al objeto, volverlo a codificar como cadena, y almacenar el nuevo JSON en SharedPreferences.

4.1.5. Viajes compartidos en bicicleta

La intención de éste proyecto es eventualmente ofrecer emparejamiento de viajes en todos los tipos de transporte. Sin embargo debido a la imperativa de distanciamiento social causada por la pandemia del COVID-19, se decidió que por ahora solo se ofrecerán viajes compartidos entre bicicletas. Ésto nos permite ir probando y optimizando el funcionamiento de los distintos componentes del sistema de emparejamiento, preparando la base del sistema para emparejar viajes en los demás tipos de transporte una vez se reanuden las actividades presenciales en la UNAM. A continuación se describen las fases del proceso para organizar un viaje compartido.

Fase 1. Búsqueda de viajes potenciales

Una vez que un usuario confirmó los detalles de su viaje, si su tipo de transporte es bicicleta antes de desplegar su ruta se le muestra una pantalla especial con un botón para activar la búsqueda de viajes compartidos (figura 4.6). Si activa la opción se le pregunta adicionalmente si ya viaja con acompañantes, ésta información no es

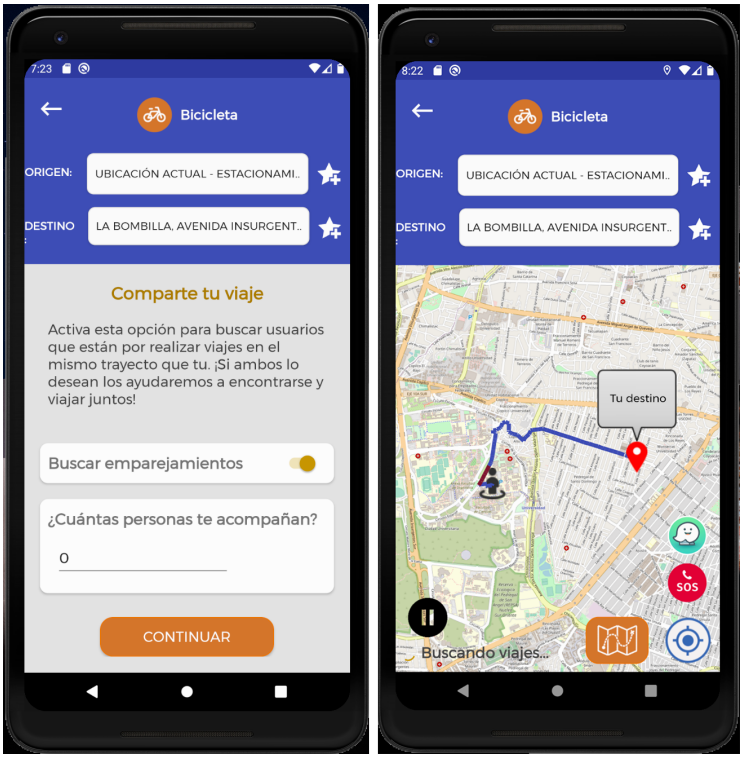

Figura 4.6: Las pantallas para activar la búsqueda de viajes compartidos, y la ruta en curso buscando emparejamientos de fondo. Imagen elaborada por el autor.

indispensable actualmente pero para viajes en auto será necesario conocer el número de asientos disponibles y necesarios.

El resto del flujo del emparejamiento ocurre en la pantalla de mapa que despliega la ruta en curso. Si se activó el emparejamiento, cada 15 segundos se realiza una consulta al servidor para ver si surgieron nuevas opciones de viaje, y aparecerá un botón nuevo en la esquina inferior izquierda, que sirve para desplegar la lista de viajes disponibles actualmente (figura 4.7), o bien para pausar la búsqueda si aún no se han hallado coincidencias (figura 4.6). Asimismo abajo de éste botón aparece un breve texto que describe el estado actual de la búsqueda (“Buscando viajes”, o “X viajes disponibles”).

Si se halló uno o más usuarios realizando viajes emparejables, el botón de pausa

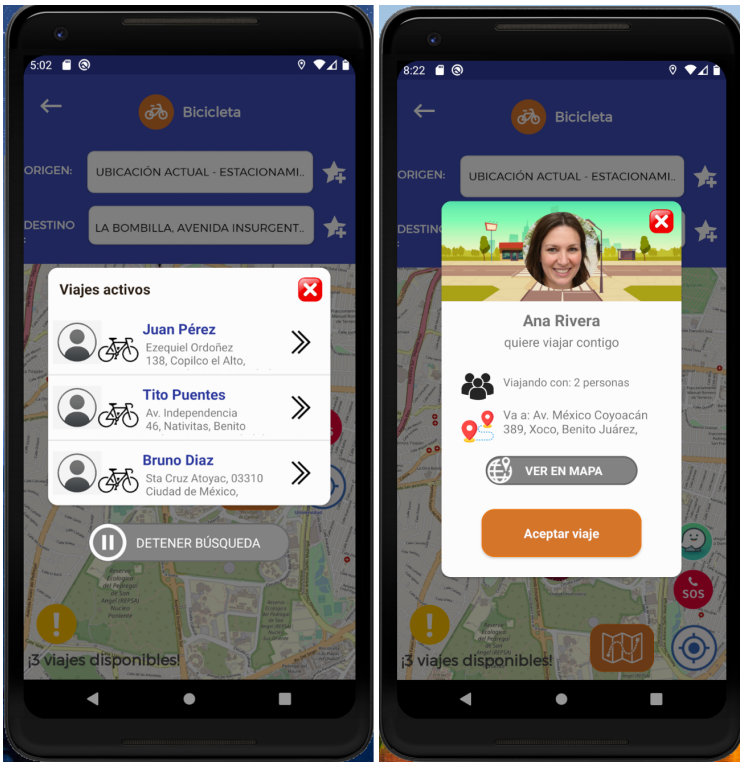

Figura 4.7: Ventanas emergentes para desplegar la lista de opciones de viajes, y para ver más detalles y confirmar un viaje potencial. Imagen elaborada por el autor.

cambia a un botón amarillo de alerta, y se envía al usuario una notificación *push* en caso de que no esté viendo la app en ese momento. Presionar el botón despliega el listado de opciones de viajes, que para cada uno muestra el nombre del usuario, su destino, tipo de transporte, y foto de perfil en caso de que exista. Seleccionar una opción de la lista muestra una tarjeta con más detalles del viaje potencial, como el número de acompañantes, el texto completo de la dirección, y contiene un botón para desplegar en el mapa la ruta propuesta (desde el punto de encuentro hasta dónde se separarían), y un botón para confirmar si se desea viajar con ese usuario.

Cuando un usuario acepta una propuesta de viaje, se le notifica de ésta acción al otro usuario mediante una conexión al servidor que se consulta periódicamente, y le aparece de inmediato una ventana emergente en la cual puede aceptarlo y ver

los detalles del viaje propuesto. Una vez que ambos usuarios confirmen el viaje se procede a la siguiente fase.

Fase 2. Ir al punto de encuentro

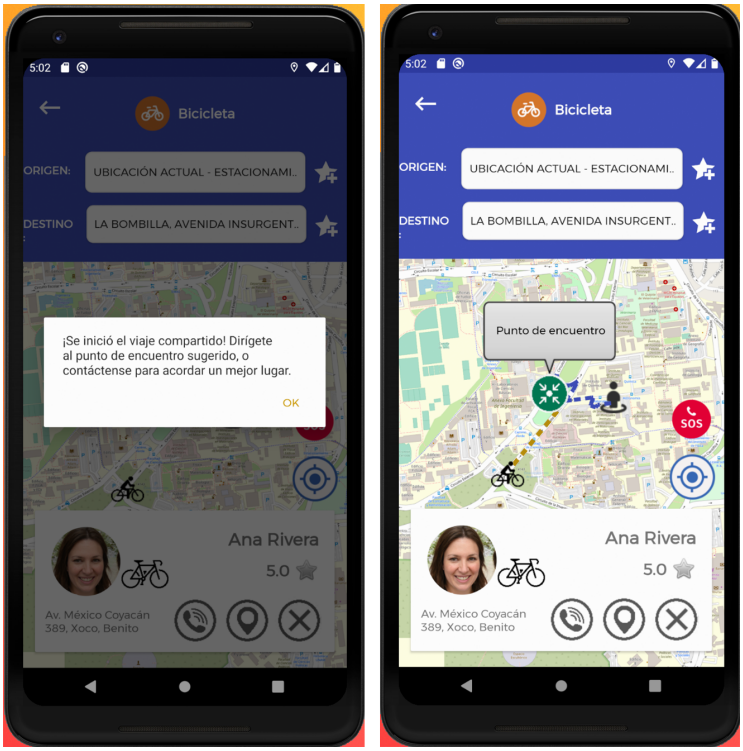

Figura 4.8: Para facilitar el emparejamiento se muestra en el mapa el punto de encuentro y la ubicación actual del potencial acompañante. El botón de emparejamiento muestra botones para llamar, confirmar encuentro, y cancelar viaje. Imagen elaborada por el autor.

Ya que dos usuarios confirmaron que desean viajar juntos, la aplicación se enfoca en ayudarlos a organizar su encuentro. Primeramente se notifica a ambos mediante un diálogo de texto que ha iniciado su viaje compartido y se les instruye que vayan al punto de encuentro. Para simplificar las instrucciones que se dan a los usuarios, en éste punto se borra del mapa la ruta completa. En su lugar se muestra la ubicación

del punto de encuentro propuesto y el camino hacia allá, así como un marcador de una persona en bicicleta que corresponde a la ubicación del otro usuario.

Los usuarios pueden verse en el punto sugerido o contactarse para organizar otro. En la zona inferior de la pantalla aparece ahora un recuadro con los principales datos del usuario emparejado, así como botones para llamarle a su número telefónico, mostrar su ubicación actual en el mapa, y cancelar el viaje compartido. Para hacer espacio para éste recuadro inferior, desaparecen el botón de emparejamiento y el botón naranja que despliega el desglose de ruta.

La aplicación automáticamente revisa la distancia entre ambos usuarios, una vez que se encuentren a 20 metros de distancia o menos se despliega un diálogo de texto que les pregunta si ya se han encontrado, en caso positivo se inicia la siguiente fase.

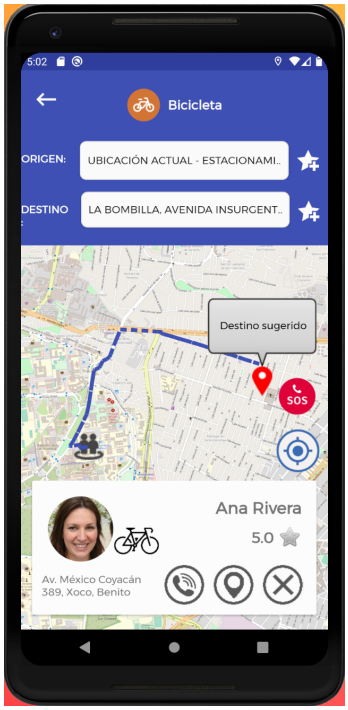

Figura 4.9: Dos usuarios viajando juntos hacia el destino propuesto. Imagen elaborada por el autor.

Fase 3. Viajando juntos.

Ya que los usuarios se encontraron, solo queda realizar el viaje (figura 4.9). En éste punto desaparece del mapa la ubicación del acompañante pues ya no es necesaria, y el marcador de la ubicación actual cambia para reflejar que viajan dos personas juntas. El marcador del destino en el mapa es ahora el punto del fin del segmento compartido que propuso el algoritmo, en donde se van a separar hacia sus respectivos destinos finales. Actualmente cuando ambos usuarios llegan a su destino el viaje simplemente concluye.

4.1.6. Consulta de recursos informativos

La aplicación contiene diversos recursos informativos para ayudar a los usuarios a tomar decisiones de movilidad en la Ciudad de México. Como la población hacia la que se dirige la aplicación son miembros de la UNAM, la mayoría de éstos sistemas ofrecen información sobre opciones de transporte en Ciudad Universitaria, así como eventos organizados por la universidad.

Pumabús en tiempo real

Gracias a una colaboración con la Dirección General de Servicios Generales y Movilidad (DGSGM) de la UNAM se instalaron transmisores GPS en todos los autobuses usados en las rutas de Pumabús en C.U. Éstas ubicaciones se transmiten al centro de control de la dirección, y nuestro servidor se conecta periódicamente al de ellos para consultar éstas localizaciones.

Los usuarios pueden acceder a éstas ubicaciones usando el botón de Pumabús de la barra inferior en el mapa principal. Para facilitar la consulta de información relevante y limitar la cantidad de elementos dibujados en el mapa, solo se muestran las rutas más cercanas a la ubicación del usuario. Al presionar el botón se dibujan en el mapa las tres paradas de Pumabús más cercanas al usuario, y una lista con

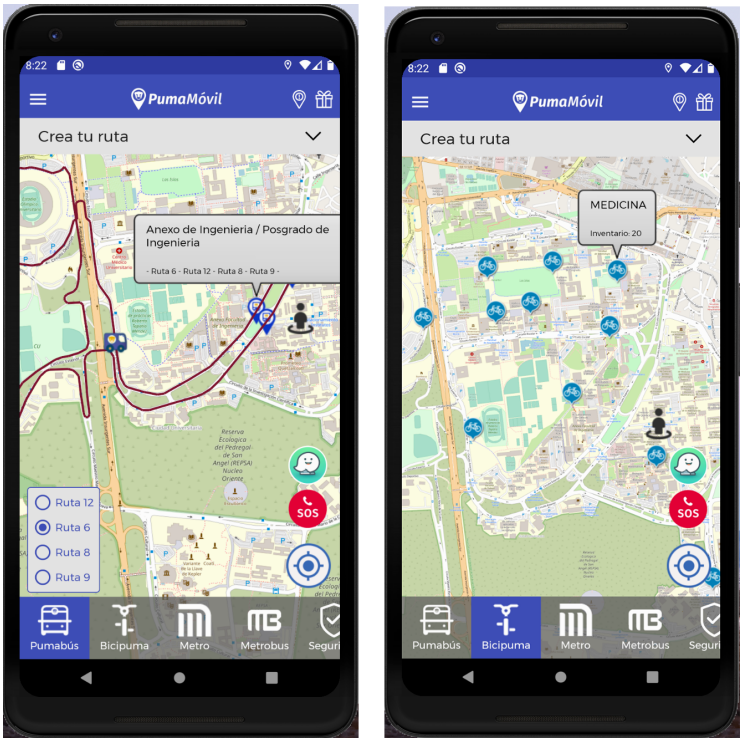

Figura 4.10: Las funcionalidades para consultar paradas y rutas de Pumabús en tiempo real, y los módulos de Bicipuma con su inventario actual, respectivamente. Imagen elaborada por el autor.

todas las rutas de Pumabús que pasan por esas paradas. Seleccionar una ruta de la lista muestra en el mapa la linea que recorre (del color correspondiente), y empieza a dibujar y actualizar la posición de todos los Pumabuses recorriendo esa ruta actualmente. Para que se pueda visualizar la mayor área del mapa posible, hacer uso de cualquiera de los botones de la barra inferior oculta la sección superior de Çrea tu ruta". Ésa sección se puede volver a mostrar y ocultar a discreción usando la flecha superior derecha.

Módulos de Bicipuma con inventario

Seleccionar el botón de Bicipuma de la barra inferior en la pantalla principal despliega en el mapa todos los módulos en C.U. donde los universitarios pueden tomar prestada una bicicleta. Presionar el marcador de una estación muestra un globo con el nombre y la cantidad de bicicletas disponibles actualmente. El inventario disponible se obtiene mediante una integración con el software usado para prestar y recibir bicicletas; gracias a la colaboración con DGSGM se implementaron funciones en las computadoras de cada módulo para que el personal pueda introducir manualmente el inventario disponible, así como registrar cada que se recibe o presta una bicicleta.

Metro y Metrobús

Para poder calcular rutas en transporte público nuestro servidor cuenta con mapas de la red del Metro y del Metrobús de la CDMX. No contamos con la capacidad de mostrar las ubicaciones de los vagones o metrobuses más cercanos, pero podemos mostrar todas las rutas y estaciones como recurso informativo.

Recursos de Seguridad y Accesibilidad en C.U.

La Dirección General de Atención a la Comunidad (DGACO) de la UNAM nos proporcionó las ubicaciones de diversos recursos de seguridad en C.U., incluyendo los postes de emergencia, módulos y bases de vigilancia UNAM, la central de atención a emergencias, y la estación de bomberos. Nuestros usuarios pueden consultar todos éstos elementos usando el botón de seguridad de la barra inferior. Asímismo, contamos con la ubicación de las rampas para discapacitados en C.U. a través del botón de Accesibilidad.

Asímismo la app cuenta con un botón S.O.S. el cual despliega un menú emergente con más acciones de seguridad y accesibilidad: con el primer botón se llama al 911 o vigilancia UNAM (el número que se marca depende de si la ubicación actual

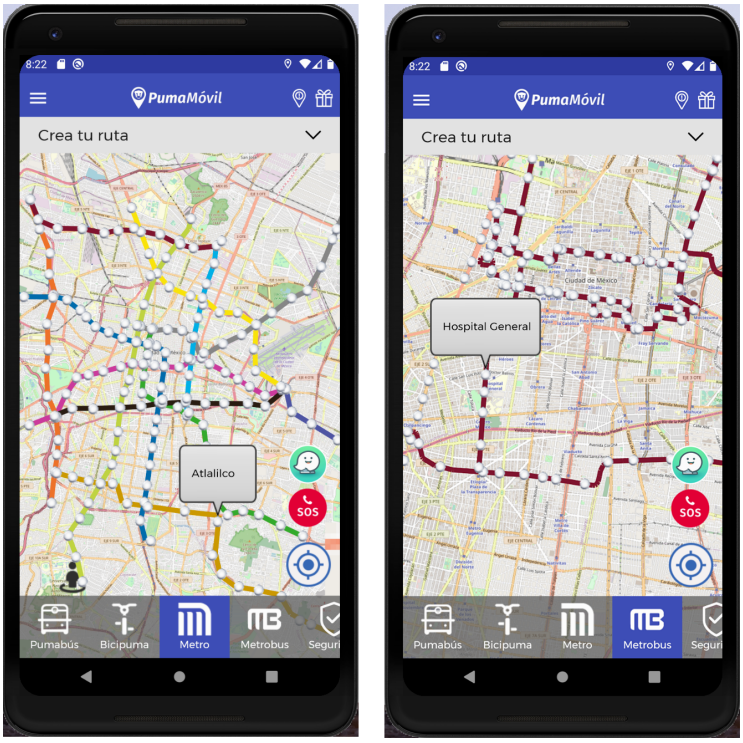

Figura 4.11: Presionar el botón de Metro y Metrobús respectivamente muestra todas las líneas y estaciones. Imagen elaborada por el autor.

del usuario se encuentra dentro de C.U.), el siguiente botón sirve para solicitar un transporte especial para gente con discapacidades que ofrece la UNAM en toda C.U., y finalmente el último permite al usuario compartir su ubicación con un contacto de su preferencia – se genera y comparte un enlace a una página de nuestro sitio web que contiene un mapa con la posición más reciente registrada del dispositivo.

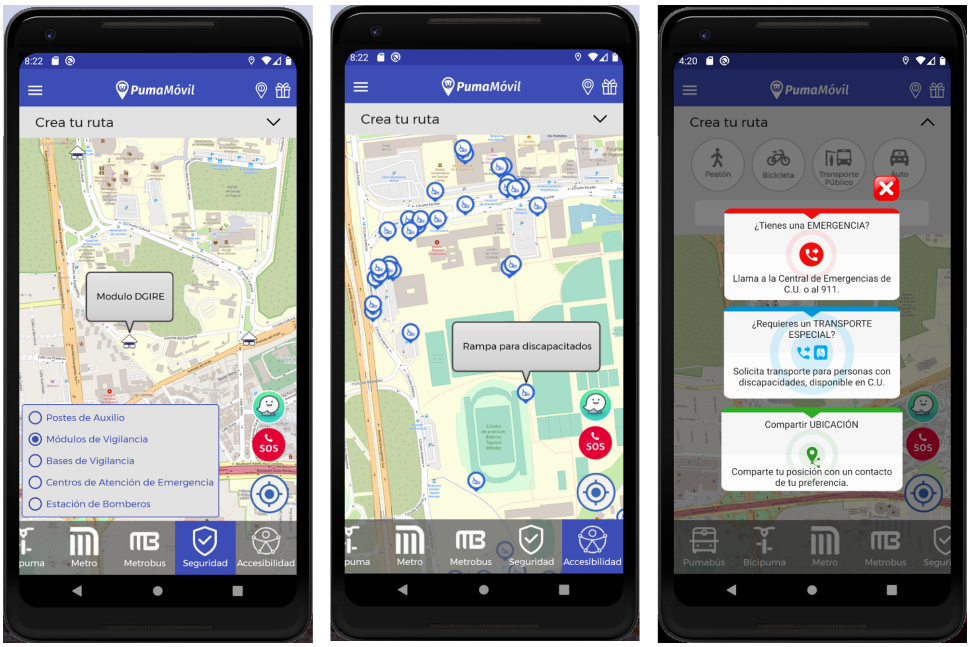

Figura 4.12: Acción de presionar el botón de Seguridad, Accesibilidad, y S.O.S, respectivamente, ubicados en la esquina inferior derecha de la pantalla principal. Imagen elaborada por el autor.

4.2. Aplicación de iOS

Mi labor en el desarrollo de la versión de PumaMóvil para iOS fue la misma que en la de Android, implementar el diseño que se acordó y darle funcionalidad conectando diversos elementos visuales con servicios REST del servidor. Ésta versión se desarrolló en Xcode en el lenguaje Swift versión 4.

Ya que las aplicaciones de Android e iOS son análogas, el funcionamiento de las pantallas a continuación se explicó en su mayor parte en las secciones respectivas sobre la app de Android, así que aquí solamente se explican las diferencias que tiene la implementación en iOS, junto con capturas de pantalla demostrativas tomadas en un simulador de un iPhone 11 con versión de iOS 13.3.

4.2.1. Pantalla principal

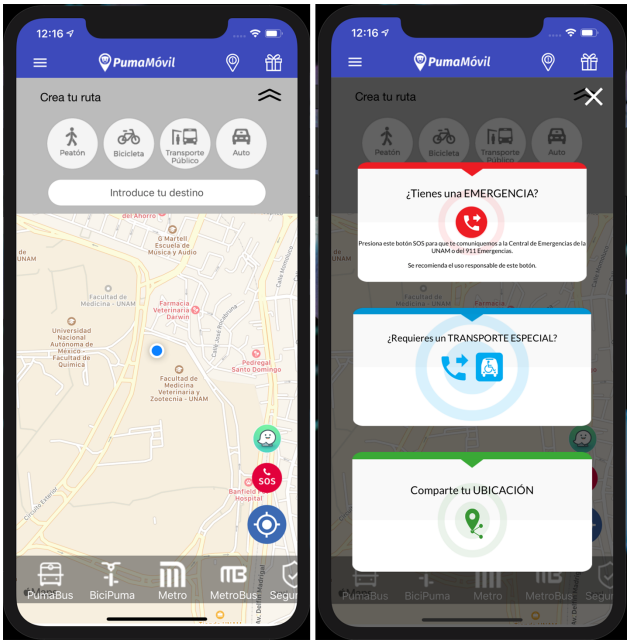

Figura 4.13: La pantalla principal y ventana emergente del botón de emergencia. Imagen elaborada por el autor.

La pantalla principal (figura 4.13) tiene el mismo diseño y funcionamiento en ambas versiones de la app. En lugar de OpenStreetMaps todos los mapas están implementados como MKMapView, que proviene de la biblioteca estándar de mapas de Apple llamada MapKit. Ya que el mapa es un componente desarrollado directamente por Apple, presenta algunas ventajas comparado con OSM. Existe una integración directa y sencilla entre los mapas de MapKit y otras funcionalidades relativas, como la barra de búsqueda de destinos y el servicio de localización GPS, ya que Apple ha implementado interfaces predefinidas entre estos sistemas; en Android estos componentes existen independientemente y las interacciones entre ellos se deben programar más explícitamente. Hemos notado que los mapas de MapKit están mejor optimizados y reaccionan con mayor rapidez, y tienen ciertos elementos visuales que no

presenta OSM, como animaciones de transición predefinidas, por ejemplo al mostrar y ocultar la barra de búsqueda, o el punto de ubicación de usuario que tiene una animación en la que pulsa periódicamente.

4.2.2. Creación de rutas

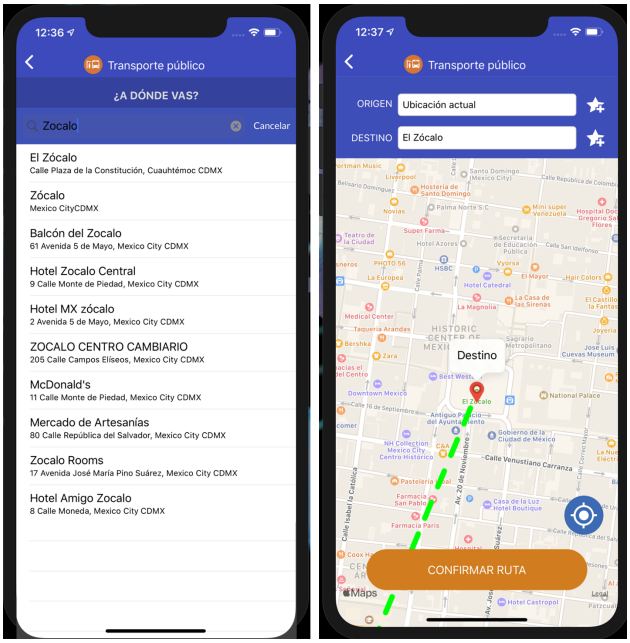

Figura 4.14: La interfaz para buscar una dirección, y pantalla de confirmación de destino. Imagen elaborada por el autor.

La biblioteca MapKit de Apple también nos proporciona funciones para buscar direcciones mediante texto, mediante la clase MKLocalSearch. Ésta clase se vincula a barras de búsqueda UISearchBar para obtener el texto a consultar y mostrar la lista de opciones. Una vez se selecciona una dirección, se muestra la ubicación en el mapa como un marcador arrastrable para que se confirme la posición exacta.

Almacenamiento de direcciones recientes y favoritas

El almacenamiento persistente de direcciones favoritas y recientes emplea la clase NSUserDefaults, que similarmente que SharedPreferences en Android permite almacenar cadenas de texto referenciadas con una llave o nombre. Así que igualmente se definió un objeto 'Direcciones', y la lista de recientes/favoritos es un arreglo de éstos objetos que se serializa como cadena de texto para su almacenamiento, usando las funciones PropertyListEncoder y PropertyListDecoder.

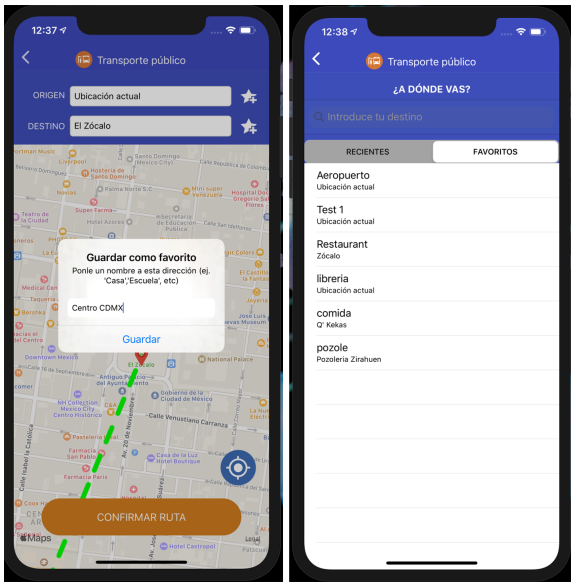

Figura 4.15: Ventana emergente para agregar una dirección favorita, y la lista de favoritos que se encuentra abajo de la barra de búsqueda de direcciones. Imagen elaborada por el autor.

Desglose de ruta en curso

Ya que se confirmó el origen y destino del viaje se dibuja en la pantalla siguiente de mapa la ruta propuesta, y el botón dorado inferior permite desplegar el desglo-

se de los pasos del viaje. El diseño de ésta pantalla contiene elementos visuales de una versión previa de la aplicación, después de que concluyan las pruebas de funcionalidad del emparejamiento en la app de Android trabajaremos en actualizar la aplicación de iOS para implementar el resto del diseño más actual, así como el flujo para crear viajes compartidos entre bicicletas.

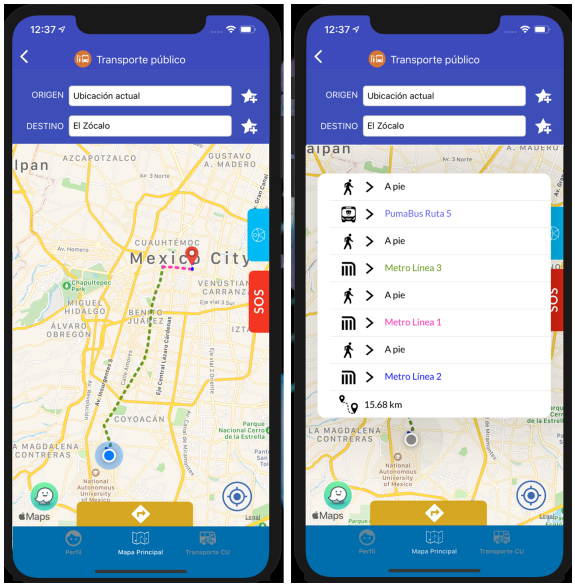

Figura 4.16: Pantalla de ruta en curso, y su ventana emergente del desglose del viaje. Imagen elaborada por el autor.

4.2.3. Consulta de recursos informativos

Pumabús en tiempo real y módulos de Bicipuma

La consulta de rutas y vehículos de Pumabús funciona de la misma manera que en Android, la única diferencia es en el diseño de la lista desplegable de rutas cercanas, en ésta versión cada ruta contiene un icono de un autobús del color correspondiente, así como una sección para mostrar el tiempo esperado de llegada del

siguiente Pumabús, que por el momento está en desuso. La sección que muestra los módulos de Bicipuma con inventario es idéntica en ambas versiones de la aplicación.

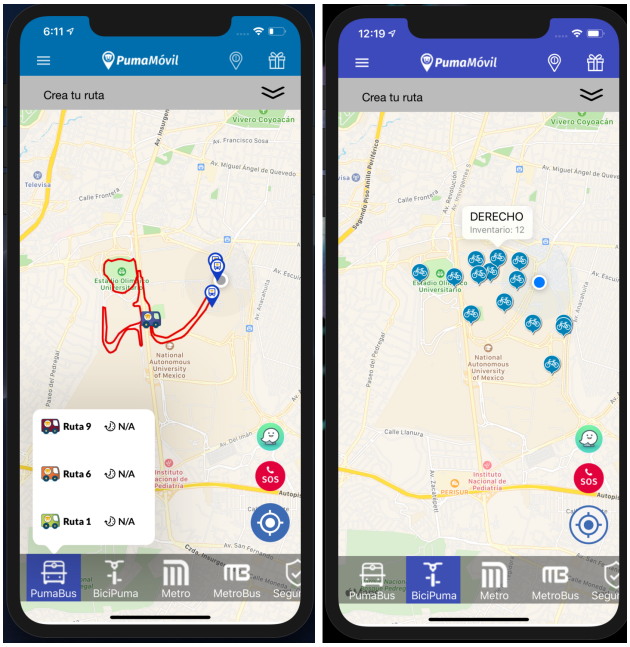

Figura 4.17: Acción de presionar el botón de Pumabús y Bicipuma respectivamente. Imagen elaborada por el autor.

Metro y Metrobús

De la misma manera en ambas versiones, los botones de Metro y Metrobus de la barra inferior dibujan en el mapa la red completa correspondiente.

Recursos de seguridad y accesibilidad en C.U.

Los últimos dos botones de la barra inferior en la pantalla principal muestran mapas de recursos de Seguridad y Accesibilidad en C.U. respectivamente, en particular los postes de emergencia, casetas de vigilancia, puntos de reunión, y rampas de accesibilidad.

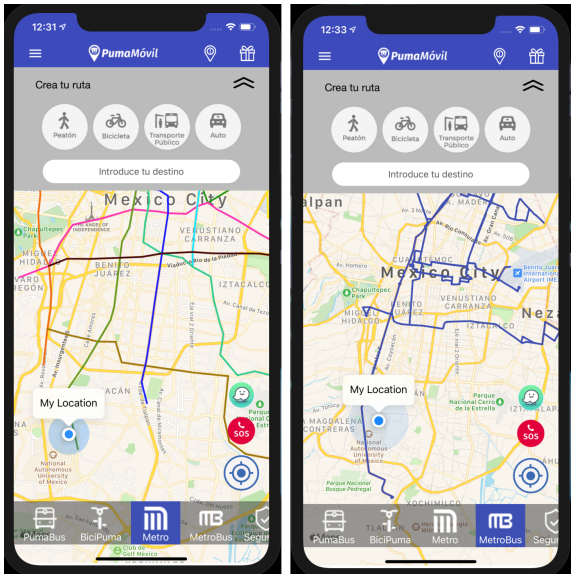

Figura 4.18: Acción de presionar el botón de Metro y Metrobús respectivamente. Imagen elaborada por el autor.

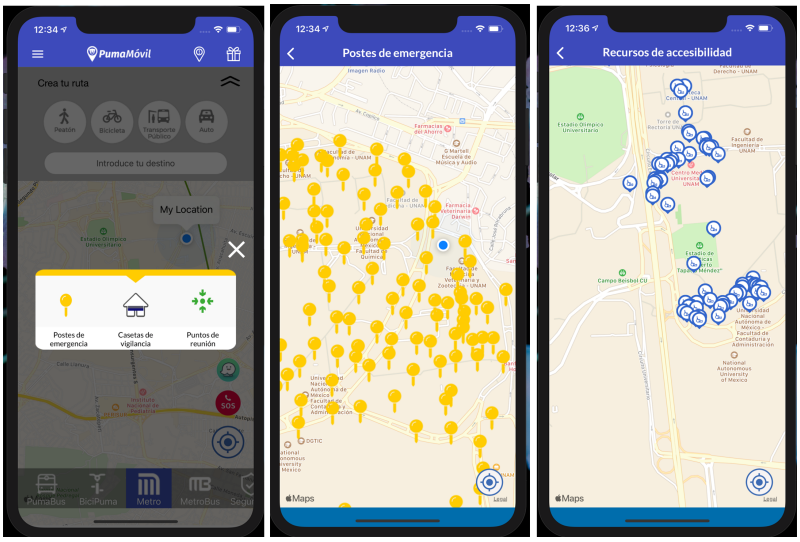

Figura 4.19: Mapas de recursos de seguridad y accesibilidad en C.U. Imagen elaborada por el autor.

4.3. Sistema de geolocalización de usuarios

Los Sistemas de Geoposicionamiento Global (GPS) presentes en todos los celulares modernos nos permite ofrecer diversas funcionalidades que dependen de la localización del usuario. Durante la creación de viajes compartidos para facilitar el encuentro es útil mostrarle a ambos usuarios sus ubicaciones mutuas, y posteriormente para dar seguimiento de viaje en tiempo real e ir actualizando la ruta restante. También se usa para mostrar elementos en el mapa que le sean relevantes al usuario, como las paradas más cercanas de Pumabús, el módulo de Bicipuma más cercano, y eventos culturales de Gaceta Digital UNAM próximos. Asímismo permite una opción en el botón S.O.S. para que el usuario comparta su ubicación en tiempo real con un contacto de su preferencia.

Los componentes de éste sistema son los dispositivos móviles generando un flujo constante de coordenadas, y un servidor central que almacene éstos datos y redirija la información conforme se requiera. Así por un lado se requirió acceder a la ubicación más reciente de usuarios en cada plataforma y enviar al servidor una actualización de la posición periódicamente, y por otro lado el servidor recibe y almacena los registros de ubicaciones en la base de datos, anonimizados y encriptados. El servidor aloja servicios REST para que usuarios registrados puedan consultar ubicaciones recientes. A continuación se detallan los componentes mencionados.

4.3.1. Modelo en base de datos

Para almacenar los registros de ubicaciones de nuestros usuarios se agregaron tres tablas a la base de datos de PostgreSQL (figura 4.20). La primer tabla es relacional y únicamente sirve para identificar a qué usuario le pertenece un dispositivo, vinculando los identificadores únicos (“ID”) correspondientes. Las otras dos tablas almacenan registros de coordenadas en un tiempo dado. Una tabla aloja el Historial de Ubicaciones de todas las posiciones comunicadas por los dispositivos; es de interés

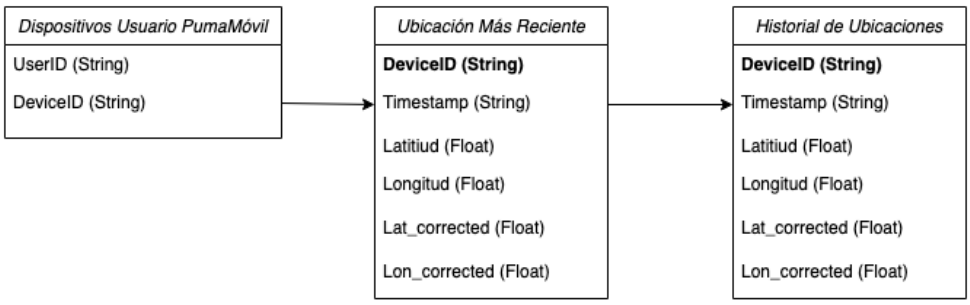

Figura 4.20: Esquema de las tablas de la base de datos usadas por éste sistema. Imagen elaborada por el autor.

conservar éste registro histórico pues será de utilidad para realizar estudios estadísticos sobre los patrones de movilidad de los usuarios. Pero consultar la posición más reciente de un dispositivo en ésta tabla ocuparía un tiempo de búsqueda cada vez mayor conforme vaya creciendo el número de renglones, por lo que se mantiene también la tabla de Ubicación Más Reciente, que contiene únicamente las últimas coordenadas recibidas de cada dispositivo. En ambas tablas cada registro contiene los siguientes elementos: el ID del dispositivo que hizo la medición, una marca de tiempo (*timestamp*) en formato ISO 8601 UTC, y ambos valores de la coordenada espacial, latitud y longitud. También se calculan las coordenadas más cercanas al usuario que caen sobre la red vial de la CDMX, usando la función de PostgreSQL $ST\_ClosestPoint$ entre las coordenadas recibidas y un mapa de la CDMX existente en la base de datos. Éstas coordenadas calculadas se almacenan bajo el nombre *lat_corrected* y *lon_corrected*, y se requieren para poder ejecutar algoritmos sobre la red vial, como el que calcula la ruta más corta entre dos puntos.

4.3.2. Servicios REST

La conexión entre los dispositivos móviles y la base de datos se realiza mediante servicios REST implementados en el Framework 'Django', el cual permite aceptar y responder conexiones HTTP (GET y POST) mediante servicios asociados a direcciones web (URLs). Para el manejo de ubicaciones de usuario se implementaron los siguientes servicios:

Nombre	Actualizar ubicación.
Acción	El dispositivo del usuario envía una actualización de su ubicación más reciente.
Parámetros de entrada	▪ Latitud (Double) ▪ Longitud (Double) ▪ Marca de tiempo ISO 8601 (String) ▪ Token de sesión de usuario (String)
Respuesta (caso exitoso)	HTTP 200 OK
Descripción	El servicio recibe las coordenadas más recientes de un usuario, se actualiza el renglón correspondiente en la tabla de Ubicación Más Reciente y se inserta en el Historial de Ubicaciones.

Servicio de ubicación 1.

Nombre	Consultar ubicación.
Acción	Una interfaz está solicitando la ubicación más reciente de un usuario.
Parámetros de entrada	▪ ID de usuario (String) ▪ Token de sesión de usuario (String)
Respuesta (caso exitoso)	HTTP 200 OK ▪ Latitud (Double) ▪ Longitud (Double)
Descripción	El servicio devuelve las coordenadas más recientes que se tengan registradas asociadas al ID de usuario dado.

Servicio de ubicación 2.

El Framework Django ofrece bibliotecas para manejar fácilmente todos los aspectos de una conexión HTTP. Por ejemplo, obviando las clases y validaciones que envuelven cada servicio, el procedimiento del Servicio 2 de ésta sección se logra con las siguientes líneas de código:

```
recent = MRC.objects.get(user_id=requested_usr_id)
return Response(
        {"detail": "SUCCESS",
        "lat":recent.latitude,
        "lon":recent.longitude},
        status=status.HTTP_200_OK,
```

```
    )
```

Donde MRC es el objeto virtual de Django que representa la tabla de MostRecentCoords en la base de datos, y requested_usr_id es un parámetro de entrada.

Asímismo, el funcionamiento del Servicio 1 de ésta sección se logra esencialmente con el siguiente fragmento de código:

```
# Actualiza ubicación en la tabla MostRecentCoords (MRC)
    recent = MRC.objects.filter(user_id=id_usr)
    if not recent: #si no hay renglón creamos uno
        recent = MRC(timestamp=timestamp, latitude=lat,
               longitude=lon, user_id=id_usr)
        recent.save()
    else: #si hay renglón se actualizan los datos
        recent = MRC.objects.get(user_id=id_usr)
        recent.timestamp = time
        recent.latitude = lat
        recent.longitude = lon
        recent.save()
# Guarda renglón nuevo en el histórico (CTO)
    obs = CTO(timestamp=timestamp, latitude=lat, longitude=lon)
    obs.save()
```

En donde MRC y CTO son los objetos virtuales que representan las tablas de MostRecentCoords y CoordsTimeObservation, el id_usr es un valor privado que se obtiene a partir del token de sesión, y los parámetros recibidos son timestamp, lat, lon, y token.

4.3.3. Lectura de ubicación en iOS y Android

Para que la aplicación pueda acceder a la ubicación del dispositivo, los usuarios deben conceder este permiso explícitamente. Para poder pedir la autorización primero es necesario declarar que se solicitará éste permiso agregando la llave 'NSLocationWhenInUseUsageDescription' en el archivo *info.plist* en el caso de iOS, o agregando el campo 'ACCESS_FINE_LOCATION' en el 'AndroidManifest.xml'. Una vez hecho ésto podemos mostrar el diálogo de texto que pregunta si se desea permitir la lectura de la ubicación, llamando la función 'requestWhenInUseAuthorization' del CLLocationManager de iOS y con 'getLastLocation' de fusedLocationClient en Android. Éste diálogo aparece la primera vez que el usuario inicia sesión.

Posteriormente el proceso que se ejecutará cada que se actualice la ubicación actual se definió implementando la función 'didUpdateLocations' en iOS y 'getLastLocation().addOnSuccessListener' en Android. Como solo accedemos a la ubicación en las pantallas que contienen mapas, éste proceso es: actualizar la posición del usuario dibujada en el mapa, actualizar los elementos dibujados en el mapa que dependan de la ubicación, y envíar una actualización al servidor mediante una conexión HTTP. En éste último paso tuvimos que considerar el uso de datos móviles, porque dependiendo de la precisión del GPS la posición del dispositivo puede actualizarse múltiples veces cada segundo; así que para evitar hacer demasiadas conexiones al servidor, simplemente se mantiene una variable con el instante de tiempo en que se envió la última medición, y solo se envía una actualización si ha transcurrido suficiente tiempo (definido por otra variable, actualmente está limitado a cada 15 segundos).

4.4. Servicios de viajes compartidos

También implementé algunas funciones en el servidor relacionadas con el sistema de viajes compartidos. En particular desarrollé los pasos que ocurren después de que el algoritmo de emparejamiento devuelve una lista de rutas semejantes: solicitar la creación de un nuevo viaje compartido, notificar al otro usuario de ésta solicitud, y aceptar un viaje compartido.

4.4.1. Modelo en base de datos

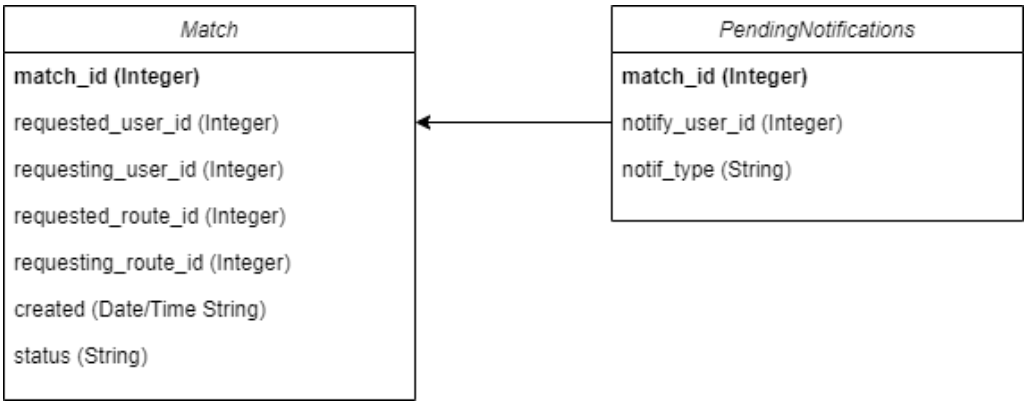

Figura 4.21: Esquema de las tablas nuevas de la base de datos. Imagen elaborada por el autor.

Para éstos servicios se agregaron dos tablas a la base de datos. El modelo Match contiene la información de todos los viajes compartidos organizados; para distinguir entre ambos, los datos del usuario que solicitó el match se denotan con el prefijo "*requesting*", y el otro usuario se denota como "*requested*". Se almacena una referencia a sus perfiles de usuario mediante el user_id, así como a la ruta original de cada uno. El campo de "status" denota la fase actual del viaje compartido, y puede ser: REQUESTED, ACCEPTED, ONGOING, ENDED, o CANCELLED. Finalmente se guarda también el instante de tiempo en que se creó el viaje.

Para el sistema de notificaciones, además de los datos del viaje emparejado propuesto (accesibles mediante el match_id) solamente se requiere saber a qué usuario se debe notificar (notify_user_id), y el tipo de notificación (notif_type), que según el evento ocurrido puede ser REQUEST, ACCEPT, END, o CANCEL.

4.4.2. Servicios REST

La funcionalidad de éste flujo se concretó en los siguientes servicios:

Nombre	Solicitar un viaje compartido.
Acción	Previamente a éste punto, el usuario creó su ruta y consultó la lista de posibles emparejamientos. El usuario selecciona una opción de la lista y presiona el botón de "Solicitar compartir viaje".
Parámetros de entrada	▪ ID de ruta actual (Integer) ▪ ID de ruta del otro usuario (Integer)
Respuesta (caso exitoso)	HTTP 200 OK
Descripción	Este servicio crea una nueva entrada en la tabla Match con los datos recibidos (y $status$ = 'REQUESTED'). Los ID de usuario se obtienen a partir de los ID de ruta dados. Asímismo se agrega un nuevo renglón en la tabla de PendingNotifications (en el cual $user\_id = requested\_user\_id$, y $notif\_type$ = 'REQUEST') indicando que se debe avisar al otro usuario de la posibilidad de viaje.

Servicio de emparejamiento 1.

Nombre	Consultar notificaciones.
Acción	Un usuario desea consultar si ha ocurrido un cambio en su situación de viaje compartido, por ejemplo: nuevas ofertas disponibles, que otro usuario solicitó compartir viaje con él, que otro usuario ha aceptado una propuesta de viaje, o si se canceló un viaje compartido.
Parámetros de entrada	▪ Token de sesión de usuario (String)
Respuesta (caso exitoso)	HTTP 200 OK ▪ Tipo de notificación (String) ▪ Datos del 'match' asociado (JSON String)
Descripción	Este servicio obtiene el ID de usuario a partir del token recibido, el cual usa para obtener de la tabla PendingNotifications todas las entradas cuyo notify_user_id coincida con el del usuario. Éstas notificaciones se serializan en formato JSON y se comunican al usuario en la respuesta a la petición HTTP. Para cada notificación se devuelve su tipo y el registro de 'match' correspondiente. Asimismo se eliminan las notificaciones enviadas de la tabla PendingNotifications para evitar notificar múltiples veces el mismo evento.

Servicio de notificaciones 1.

Nombre	Aceptar un viaje compartido.
Acción	Previamente a éste punto, el usuario recibió una notificación indicando que otro usuario desea que compartan viaje. El usuario presiona el botón de "Aceptar Viaje".
Parámetros de entrada	▪ Token de sesión de usuario (String) ▪ Match ID (Integer)
Respuesta (caso exitoso)	HTTP 200 OK
Descripción	Este servicio valida que el usuario forme parte del 'match' especificado por el Match ID, luego cambia el valor de la variable *status* del 'match' a 'ACCEPTED', y agrega una nueva entrada en PendingNotifications para indicar al otro usuario que se puede iniciar el viaje ($notif\_type$ = 'ACCEPT').

Servicio de Emparejamiento 2.

Nombre	Actualizar *status* de un viaje.
Acción	Un usuario confirma que ha encontrado al otro usuario emparejado y se va a iniciar la siguiente fase del viaje, o bien un usuario indica que ha concluido el viaje.

Parámetros de entrada	▪ Token de sesión de usuario (String) ▪ Match ID (Integer) ▪ Tipo de actualización (String)
Respuesta (caso exitoso)	HTTP 200 OK
Descripción	Este servicio valida que el usuario sea parte del 'match' dado, de ser así cambia el *status* del 'match' a 'ONGOING' o 'ENDED' según la acción indicada en el parámetro de entrada, y agrega la notificación correspondiente en la tabla PendingNotifications ($notif\_type$ = 'ENDED').

Servicio de Emparejamiento 3.

Nombre	Cancelar un viaje compartido.
Acción	Un usuario solicita cancelar su emparejamiento actual.
Parámetros de entrada	▪ Token de sesión de usuario (String) ▪ Match ID (Integer)
Respuesta (caso exitoso)	HTTP 200 OK

Descripción	Este servicio valida que el usuario sea parte del 'match' dado, de ser así cambia permanentemente el *status* del 'match' a 'CANCELLED', y agrega la notificación correspondiente en la tabla PendingNotifications ($notif\_type$ = 'CANCEL').

Servicio de Emparejamiento 4.

Capítulo 5

Evaluación del producto

Ésta sección contiene una evaluación de distintas características de la app que son frecuentemente usadas en el ámbito de la Ingeniería de Software como Métricas Estándares de Calidad de Software. En cada caso se explica brevemente cómo se atendió dicho requerimiento en la implementación final.

Además de los requerimientos establecidos por los estándares de calidad de software y los responsables del proyecto, antes de poder publicar una app en las tiendas de Play Store y App Store se requiere que la app supere una serie de pruebas automatizadas. Éstas pruebas validan aspectos como su estabilidad (que el software no cierre inesperadamente ante distintos escenarios), portabilidad (que la app funcione en todos los dispositivos que soporta), y accesibilidad (que el tamaño de fuente sea legible, y que todos los botones sean compatibles con la función de descripciones auditivas para invidentes). La versión 1.0 de nuestra aplicación superó exitosamente las pruebas automatizadas de ambas plataformas y se encuentra disponible en las tiendas. Adicionalmente, nuestra implementación respeta las Pautas de Diseño y de Accesibilidad para desarrolladores establecidas por Google y Apple [65, 67, 66, 68].

Por otro lado, con el fin de unificar la apariencia de aplicaciones institucionales la DGTIC establece lineamientos para el desarrollo de aplicaciones móviles de la

UNAM. Dichos requerimientos incluyen: una pantalla de bienvenida (splash screen) que muestre durante tres segundos el escudo de la UNAM, que los encabezados deben incorporar los colores oficiales de la universidad (azul marino y oro) así como el escudo de la UNAM en caso de aplicaciones de escritorio, y se debe demostrar que la aplicación cumple con las medidas especificadas por DGTIC de seguridad y privacidad de uso de datos personales. La versión 1.0 de nuestra aplicación cumplió los requerimientos establecidos exitosamente, lo cual le permite estar listada como una aplicación móvil oficial de la UNAM en http://apps.unam.mx/.

5.1. Requerimientos funcionales

Los requerimientos funcionales de un producto de software se refieren a las acciones que el software debe realizar para funcionar exitosamente. En nuestro caso esta especificación de requerimientos fue establecida previamente al inicio del desarrollo por los responsable del proyecto. Concretamente, los requerimientos funcionales de las aplicaciones de Android e iOS son que sus interfaces deben servir para realizar las siguientes operaciones y consultas en el servidor de la aplicación: registro de nuevos usuarios con confirmación por correo, inicio de sesión, consultar la ruta más corta en cuatro tipos de transporte, asignación de viajes emparejados en bicicleta, y visualizar diversos recursos informativos en mapas. Tanto la aplicación de iOS como la de Android fueron probadas y evaluadas por los responsables del proyecto, quienes han validado que se cumplieron exitosamente todos los requerimientos funcionales solicitados.

5.2. Requerimientos no funcionales

5.2.1. Usabilidad

La usabilidad del software se refiere a qué tan intuitivo y sencillo le parece su uso a la mayor cantidad de gente posible. Para optimizar éste aspecto nos hemos auxiliado de patrones de diseño, soluciones establecidas para resolver problemas que surgen frecuentemente durante el desarrollo de software. Se aplicaron patrones de diseño provenientes de diversas fuentes [64, 69, 70] relativos a la Experiencia de Usuario ('User Experience' o UX) y al diseño de interfaces de aplicaciones móviles, así como las Pautas de Diseño y de Accesibilidad establecidas por Google y Apple [65, 67, 66, 68]. Los principales patrones de diseño relativos a usabilidad que sigue nuestro software son los siguientes:

- **Pantalla de bienvenida (*Splash screen*)**. Es una pantalla que se muestra brevemente al iniciar la app. Su propósito principal es permitir tiempo para que cargue la app, así como reforzar la identidad de marca (brand identity) y empezar a establecer la temática visual. Nuestra app contiene dos pantallas de bienvenida que muestran el escudo de la UNAM y el logotipo de la app respectivamente (figura 5.1), las cuales duran 3 segundos y se muestran una después de otra.

- **Pantallas introductorias (*Onboarding screens*)**. Se refieren a una sección tutorial para que los nuevos usuarios conozcan las funcionalidades de la app y cómo usarlas. En nuestro caso se implementó un carrusel de imágenes (figura **??**) que los usuarios pueden consultar en cualquier momento en la sección de Ayuda del menú lateral, adicionalmente la primera vez que se inicia sesión se muestra un diálogo emergente preguntando al usuario si desea abrir el tutorial.

- **Pantalla principal (*Home screen*)**. Al ser la interfaz central de la apli-

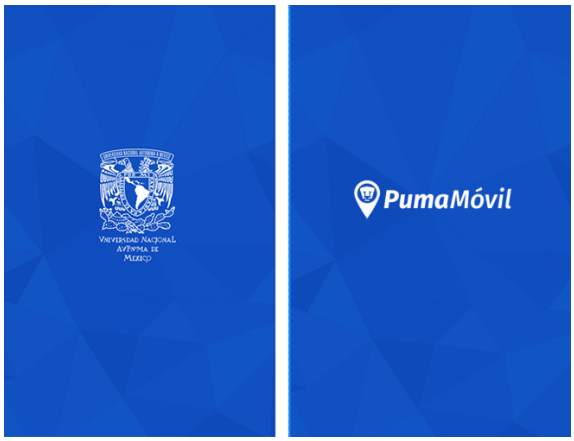

Figura 5.1: Pantallas de bienvenida a la app. Imagen elaborada por el autor.

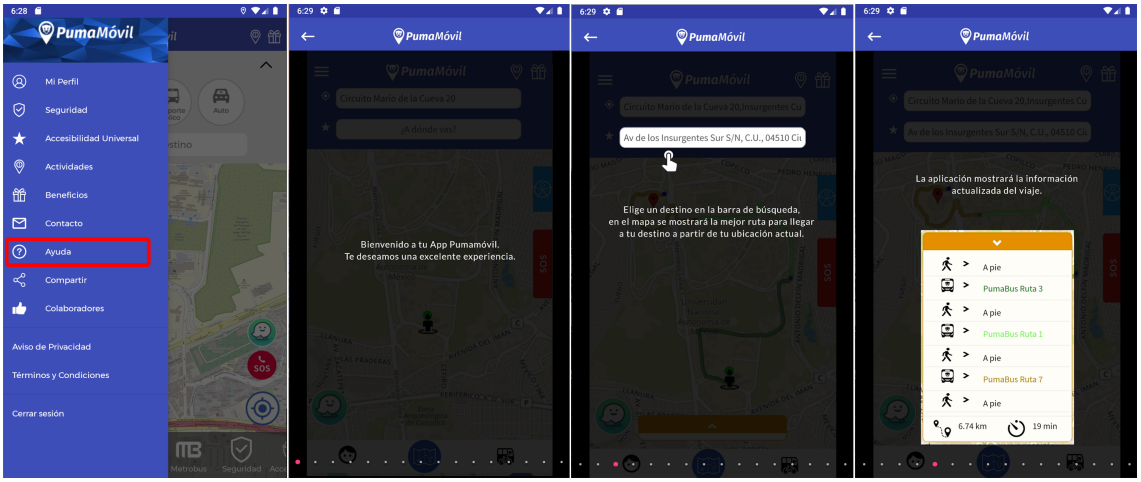

Figura 5.2: Tutorial para primeros usuarios, ubicado en la sección de Ayuda del menú lateral (resaltada en la primer pantalla). Imagen elaborada por el autor.

cación, es importante que la pantalla principal tenga una distribución de elementos clara e intuitiva, que permita acceder rápidamente a todas las funcionalidades de la aplicación. El diseño de ésta interfaz se analiza a detalle en la sección 4.1.3 de ésta tesis.

- **Registro e ingreso de usuarios**. Para reducir la tasa de abandono de la app es importante que éstas secciones sean tan rápidas y sencillas como sea posible, ya que la mayoría de los usuarios encuentran éste proceso aburrido. Nuestro registro consiste en solo dos pasos, llenar el formulario correspondiente (estudiante/trabajador/otro) y activar la cuenta mediante una confirmación por correo electrónico. Asímismo los formularios fueron diseñados para solicitar la mínima información que sea indispensable tener de los usuarios. Posteriormente el ingreso únicamente requiere correo y contraseña.

- **Ventanas emergentes (*pop-ups*)**. Los patrones de diseños de ventanas emergentes buscan evitar que éstas alertas se sientan invasivas o molestas: indican que cada una debe surgir en un solo punto específico de la app, y debe mostrar controles e información relevante a esa acción en particular. Además si no es un mensaje de diálogo (aquellos que solo esperan que el usuario responda Sí/No) el contenido original de la app debe permanecer visible en el fondo. La aplicación respeta éstos lineamientos, las ventanas emergentes que no son diálogos de confirmación consumen espacio del borde de la pantalla, permitiendo ver e interactuar con el mapa y demás botones de la interfaz. Dichas ventanas emergentes incluyen el desglose de pasos de una ruta activa (figura 4.4), la lista de rutas de Pumabús (figura 4.10), la lista de recursos de seguridad (figura 4.11), y la tarjeta inferior durante un viaje compartido (figura 4.8). Los dos primeros ejemplos mencionados se pueden ocultar presionando su botón asociado.

- **Animaciones**. Las animaciones son un aspecto que mejora sutilmente la experiencia de los usuarios al hacer que el software se sienta más interactivo y reactivo. Los elementos animados en nuestra app incluyen redimensionar los mapas para que todos los marcadores sean visibles cuando se agregan elementos nuevos, *loading spinners* que aparecen cuando se espera la respuesta

de una petición HTTP, así como animaciones de transición predefinidas por Android e iOS. Asímismo pueden servir para comunicar información tácitamente al usuario, por ejemplo si en la pantalla principal se intenta introducir un destino sin haber seleccionado un tipo de transporte, los cuatro botones parpadean en naranja para llamar la atención del usuario a esa acción.

- **Accesibilidad**. Además de los lineamientos de accesibilidad requeridos para la publicación de la app en las tiendas de Android e iOS, se implementaron los siguientes recursos específicamente para visitantes de Ciudad Universitaria con necesidades de accesibilidad: consultar la ubicación en el mapa de todas las rampas de accesibilidad en C.U., y un botón para marcar al número del Transporte Especial en C.U., que permite solicitar un transporte gratuito adecuado para sillas de ruedas en cualquier lugar de C.U 4.12.

5.2.2. Portabilidad

Considerando que una gran parte de la población objetivo de usuarios son estudiantes que tienen una variedad de celulares económicos, ambas aplicaciones son compatibles con la mayoría de dispositivos móviles.

La aplicación de Android funciona en cualquier versión del SDK igual o mayor a la 15 (“Ice Cream Sandwich”), lo cual según las estadísticas que ofrece Google Play Store abarca a 99.8 % de todos los usuarios de Android. La aplicación en iOS soporta iPhones y tabletas corriendo una versión de iOS 10.0 o superior, lo cual abarca a 99.72 % de todos los usuarios.

En ambas versiones el diseño de pantallas es responsivo y se adapta a diversos tamaños de pantalla incluyendo tabletas, solamente se requiere que el dispositivo cuente con GPS.

5.2.3. Seguridad

Además de los protocolos estándar de seguridad que se requieren para distribución en Play Store y App Store relativos al firmado de la aplicación y el manejo de datos personales, se implementaron procedimientos para garantizar que solo usuarios registrados puedan hacer uso de los servicios disponibles. Todas las conexiones al servidor requieren que el encabezado de la petición contenga un token de autenticación de sesión, que es una cadena aleatoria de 20 caracteres enviada por el servidor cuando el usuario inició sesión; sin un token válido correspondiente a una sesión activa el servidor niega acceso a la mayoría de las consultas, como protección ante ataques de tipo DDOS.

El proceso de registro de usuarios emplea una autenticación de dos factores, pues para poder activar una cuenta nueva se requiere acceder a un enlace enviado por correo electrónico. Asimismo todos los datos personales se almacenan encriptados en tanto el dispositivo como el servidor, y el servidor de base de datos contiene restricciones de seguridad adicionales, reservadas por privacidad.

5.2.4. Estabilidad

Todos los productos de software son susceptibles a 'bugs' o errores inesperados en el comportamiento, y usualmente se clasifican según su grado de severidad. Los errores más críticos son los que ocasionan 'crashes' y detienen por completo la aplicación. Después están los que impiden el funcionamiento de algún componente del software, y su severidad depende de qué tan esencial sea la acción que falló. Finalmente hay errores menores que no interfieren con la funcionalidad esencial pero afectan la experiencia y claridad del usuario, por ejemplo tener textos en blanco, botones mal posicionados, etc.

El lanzamiento de la primer versión de la aplicación nos permitió identificar errores en todas las categorías, y actualmente se encuentran corregidos todos los errores

severos que se han detectado. Los errores más graves en ambas aplicaciones estaban relacionados a dos cuestiones. Durante la transición entre pantallas o actividades, algunos dispositivos Android con poca memoria RAM crasheban al tratar de cargar imágenes muy grandes, naturalmente se solventó reduciendo la resolución de todas las imagenes. La otra fuente de fallos en tanto Android como iOS era por un manejo inadecuado de hilos de ejecución que modifican un mismo mapa: muchos elementos que se dibujan en el mapa dependen de conexiones asíncronas al servidor, esto ocasiona problemas si la respuesta del servidor llega mientras se está modificando el mapa en otro hilo. La sencilla solución fue implementar variables booleanas de 'semáforo'. Incialmente su estado es 'False', cuando alguna función empieza a modificar el mapa se cambia manualmente el estado del semáforo a 'True'; así todas las funciones que modifican el mapa pueden revisar el semáforo antes, y si indica 'True' solo deben consumir tiempo hasta que el semáforo cambie de vuelta.

5.2.5. Tasa de entrega

Después del lanzamiento de la primer versión de la aplicación los usuarios identificaron diversos bugs y problemas de usabilidad, por lo que se dedicó un periodo de tiempo para implementar las correcciones necesarias. Posteriormente al lanzamiento de la versión 1.0 se publicaron diez actualizaciones en tanto Android como iOS, atendiendo problemas relativos a la funcionalidad del registro, flujo de pantallas, legibilidad de botones y geolocalización, entre otras. La tasa de entrega fue adecuada para resolver los asuntos hallados. La publicación de nuevas versiones se ha visto afecta por la pandemia causada por el COVID-19.

5.2.6. Eficiencia

La aplicación no realiza muchos cálculos o procedimientos que requieran un procesamiento local intenso, por lo que los mayores tiempos de espera son debidos a

consultas de servicios remotos. Aún así, el tiempo de respuesta es usualmente rápido, y la mayoría de las conexiones HTTP tienen un tiempo de respuesta menor a 5 segundos. Los servicios de trazado de ruta y búsqueda de viajes compartidos demoran entre 5 y 15 segundos usualmente.

En algunos dispositivos se demora el trazado del mapa la primera vez que se abre la app, ya que se deben descargar los segmentos del mapa desde servidores de OpenStreetMaps. Pero una vez que se descargan segmentos éstos se almacenan en la memoria local del dispositivo, lo que mejora sustancialmente la velocidad de carga en usos subsecuentes.

5.2.7. Mantenibilidad

Ya que éste proyecto involucra la participación de varios programadores fue importante mantener una documentación adecuada de todos los sistemas desarrollados. Todas las funciones del código contienen comentarios explicativos sobre los parámetros que recibe y devuelve, y se explican ciertos detalles del procesamiento que se realiza. Se implementaron mecanismos en Android e iOS para que cuando ocurra un fallo, se envíe automáticamente un reporte técnico al servidor indicando la versión del sistema operativo, tipo de dispositivo, pantalla donde surgió el error, y el *stack trace* que describe el problema. Asímismo contamos con documentos internos de diagramas de los sistemas, los parámetros de conexión al servidor y la base de datos, y la especificación de todas las conexiones que ofrece el servidor, incluyendo la URL de acceso, tipo de conexión (GET/POST), parámetros de entrada, salida, y encabezado de autenticación.

Capítulo 6

Lanzamiento

Desafortunadamente el lanzamiento de la versión de la aplicación con viajes compartidos en todos los tipos de transporte se vio postergado indefinidamente debido a la pandemia causada por el COVID-19. En marzo de 2020 la UNAM decidió suspender todas las actividades presenciales en todos los planteles, restricción que continúa a la fecha, entonces simplemente no existen suficientes usuarios activamente realizando viajes como para poder probar el sistema de emparejamiento. Más aún, no creemos apropiado incentivar situaciones con riesgo de contagio, como lo es compartir un automóvil entre desconocidos. Por éste motivo se redirigió el enfoque del proyecto al actual, que ofrece viajes compartidos únicamente entre bicicletas. De ésta manera podremos probar y iterar sobre la funcionalidad de emparejamiento en un transporte que permita mantener una distancia segura entre usuarios. El lanzamiento de ésta versión está programado para agosto del 2021, y esperamos tener la participación voluntaria de varios grupos ciclistas para la primera fase piloto.

En el año anterior a la pandemia se publicó la primera versión de la aplicación en tanto la Play Store de Android como la App Store de iOS. Ésta versión cuenta con las funcionalidades para consultar la ruta más corta hacia un destino en la ZMVM, las rutas de Pumabús y la posición más reciente de Pumabuses en tiempo

real, los módulos de Bicipuma, recursos informativos de seguridad y accesibilidad, y el botón S.O.S. El propósito del lanzamiento de esta primera versión era comprobar si las herramientas para consultar la ubicación de Pumabuses en tiempo real y los módulos de Bicipuma son suficientes para atraer y retener usuarios. En éste aspecto la primer versión fue exitosa. A la fecha tenemos 12,484 usuarios registrados, y en los meses posteriores al lanzamiento había alrededor de 500 usuarios diarios consultando la aplicación. Como se esperaba la ubicación en tiempo real de Pumabuses es la funcionalidad más consultada, la mayoría de las descargas fueron de la versión de Android, y corresponden a alrededor de 91 % de todos los usuarios.

Capítulo 7

Trabajo futuro

Habiendo implementado los sistemas que constituyen el núcleo la aplicación – cuentas de usuario, creación y emparejamiento de viajes – se tienen planeadas varias adiciones con el fin de incrementar la retención de usuarios. Asimismo se han pensado distintos posibles usos de la aplicación en un mediano a largo plazo. Se mencionan a continuación en el orden aproximado que se tiene planeado para su implementación.

Por un lado, una vez que se tengan suficientes usuarios es de interés realizar estudios de patrones de movilidad usando los datos anonimizados de ubicaciones geográficas. Ésto permitirá estudiar cómo se distribuyen espacialmente los visitantes de Ciudad Universitaria según el horario y día de la semana, lo cual ayudará a identificar zonas de alto tráfico y planificar intervenciones adecuadas. Asímismo nos permitirá ver si existen o no cambios en los hábitos de movilidad de los usuarios como consecuencia del uso de la aplicación, así como cuantificar el comportamiento espacial de los usuarios usando métricas de ciencia de redes, para determinar si efectivamente ésta intervención incrementa la presencia de comportamiento complejo en el sistema en general.

Respecto a funcionalidades de la aplicación, primeramente debemos implementar el sistema de viajes compartidos en los cuatro tipos de transporte que ofrecemos,

automóviles, bicicletas, transporte público, y a pie. Ésto involucra algunas consideraciones nuevas, ya que entre bicicletas el emparejamiento es simétrico o bidireccional, pero con automóviles debemos distinguir entre conductores (ofertas) y pasajeros (demandas); no se pueden emparejar conductores entre sí, pero los pasajeros – que pueden haber escogido viajar a pie o en transporte público – sí se podrían emparejar mutuamente si desean viajar acompañados. Asímismo debemos solicitar información adicional en el caso de conductores, principalmente su licencia de conducir, así como el modelo y placa de su vehículo.

Hemos planeado varios sistemas auxiliares a la creación de viajes compartidos que serán relativamente sencillos de implementar. Para los usuarios que no deseen compartir su número telefónico con sus emparejamientos, queremos implementar un sistema interno de mensajes de texto, disponible solamente para dos usuarios que han aceptado un viaje compartido juntos. Por otro lado para facilitar la confianza mutua entre posibles parejas creemos importante un sistema de calificaciones entre usuarios, en el cual al finalizar un viaje compartido los viajeros pueda enviar una calificación (1-5 estrellas) y comentario de texto, para que otros usuarios puedan consultar el historial de reseñas de sus parejas potenciales. Otra adición planeada para tener mayor validación de usuarios es vincular la cuenta con otras redes sociales como Facebook, Twitter, e Instagram, donde cada uno se mostraría como un emblema en el perfil ("Vinculado con X").

Posteriormente planeamos un sistema más elaborado pero que sería de gran utilidad para retener usuarios frecuentes, este consiste en una "ludificación" (*gamification*) de la aplicación. En resumen la idea es ofrecer premios y beneficios a los usuarios para incentivar el uso de distintas secciones de la aplicación. Los usuarios podrán acumular puntos al hacer diversas acciones como: concluir un viaje compartido, crear una nueva oferta de viaje, consultar recursos informativos, o incluso la primer vez que se abra la aplicación cada día. La cantidad de puntos otorgados será balanceada dependiendo de qué tanto se requiera incentivar cada acción, así

como la cantidad de premios que haya disponibles. Por ejemplo se otorgará una cantidad muy grande de puntos a los conductores que ofrezcan su auto y completen viajes compartidos, pues anticipamos que éste será el grupo de usuarios que más se necesita. Las recompensas a ofrecer aún no se han definido pues se deben negociar colaboraciones entre el C3 y otras entidades, pero se están investigando propuestas para premios mayores como accesos a espacios exclusivos de estacionamientos controlados en C.U., descuentos en ciertas actividades culturales o eventos de la UNAM, y descuentos en empresas que tengan convenios con la UNAM.

Finalmente, una vez que se haya mostrado la viabilidad de la aplicación en la comunidad de miembros de la UNAM el paso siguiente es exportar el sistema de creación de viajes compartidos a otras comunidades. Queremos intentar implementar un sistema para cualquier usuario de la CDMX o incluso la ZMVM, ya que es interesante la pregunta de si es posible alcanzar una masa crítica de usuarios con orígenes y destinos más heterogéneos y distribuidos en un territorio extenso, así como si los usuarios la usarían aún sin tener la confianza de estar validados por la misma universidad. En éste caso sería extremadamente importante implementar más mecanismos de validación de identidad, por ejemplo para corroborar el INE o permiso de conducir usando una base de datos oficial del gobierno. Por supuesto, también existe la posibilidad de exportar el sistema a comunidades más pequeñas como empresas u otras universidades, en las que se pueden validar los usuarios usando sus propias bases de datos.

Capítulo 8

Conclusión

Con base en la teoría de sistemas complejos y en particular las propiedades auto-organizantes de sistemas urbanos, se desarrolló una aplicación tecnológica novedosa para mejorar la movilidad urbana mediante un sistema automatizado para la organización de viajes compartidos dinámicos, así como con sistemas de información en tiempo real. Aunque el sistema completo de viajes compartidos fue postergado por la pandemia del COVID-19, el lanzamiento de la primer versión de la aplicación indica que tan solo la herramienta para consultar la ubicación de Pumabuses en tiempo real es suficiente para atraer a miles de usuarios, una cantidad suficiente para tener una masa crítica de usuarios, según algunos modelos computacionales [56]. Se espera que las nuevas herramientas informativas de la versión próxima, y el mismo sistema de viajes compartidos, logren aumentar aún más el número de usuarios recurrentes.

Mis contribuciones al desarrollo del proyecto involucraron la implementación de interfaces de las aplicaciones de Android e iOS, modelos en la base de datos, y servicios HTTP en el servidor de aplicación. Éstas contribuciones requirieron que aplicara diversos conceptos aprendidos en la carrera de Ciencias de la Computación. En particular me fueron de utilidad conocimientos adquiridos en las siguientes ma-

terias. Ingeniería de Software me enseñó cómo llevar una documentación adecuada para un proyecto colaborativo de software, me ayudó a integrarme fácilmente a un equipo 'ágil' de desarrollo de software, y aprendí diversos mecanismos de control de calidad y detección de errores. Fundamentos de Bases de Datos me permitió modelar, implementar y optimizar las tablas SQL que necesité agregar al sistema. Estructuras de Datos me permitió programar cómodamente usando estructuras como arreglos, listas ligadas, y diccionarios. Las materias de Análisis de Algoritmos y Complejidad Computacional me permitieron comprender y cuantificar el desempeño de los algoritmos de trazado de ruta más corta y emparejamiento de viajes. En las materias de Modelado y Programación, y Diseño de Interfaces de Usuario, aprendí buenas prácticas de programación a través de los patrones de diseño relativos a programación orientada a objetos, bases de datos, e interfaces móviles. Criptografía y Seguridad me ayudó a comprender los mecanismos de seguridad de la app, como la autenticación por dos factores en el registro de usuarios, los tokens para validar sesiones activas, y la encripción de datos para su almacenamiento. Finalmente Computación Concurrente me permitió identificar errores de sincronización causados por condiciones de carrera en llamadas HTTP simultáneas, y corregirlos implementando semáforos binarios.

Capítulo 9

Referencias bibliográficas

[1] T. Reed, "Inrix global traffic scorecard," 2019.

[2] Instituto Mexicano para la Competitividad, "Movilidad segura para todos." `https://imco.org.mx/movilidad-segura-para-todos/`, 2020.

[3] M. L. Bell, D. L. Davis, N. Gouveia, V. H. Borja-Aburto, and L. A. Cifuentes, "The avoidable health effects of air pollution in three Latin American cities: Santiago, Sao Paulo, and Mexico City," *Environmental research*, vol. 100, no. 3, pp. 431–440, 2006.

[4] C. Gershenson, "Living in living cities," *Artificial life*, vol. 19, no. 3_4, pp. 401–420, 2013.

[5] M. A. Bedau, J. S. McCaskill, N. H. Packard, and S. Rasmussen, "Living technology: Exploiting life's principles in technology," *Artificial Life*, vol. 16, no. 1, pp. 89–97, 2010.

[6] C. Gershenson, "Self-organization leads to supraoptimal performance in public transportation systems," *PLoS One*, vol. 6, no. 6, p. e21469, 2011.

[7] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in applied self-organizing systems*, pp. 45–55, Springer, 2013.

[8] INEGI, "Encuesta Origen Destino en Hogares de la Zona Metropolitana del Valle de México (EOD) 2017," 2017.

[9] G. Wilson-Doenges, "An exploration of sense of community and fear of crime in gated communities," *Environment and behavior*, vol. 32, no. 5, pp. 597–611, 2000.

[10] C. Gershenson and F. Heylighen, "How can we think the complex," *Managing organizational complexity: philosophy, theory and application*, vol. 3, pp. 47–62, 2005.

[11] B. Beckage, S. Kauffman, L. J. Gross, A. Zia, and C. Koliba, "More complex complexity: Exploring the nature of computational irreducibility across physical, biological, and human social systems," in *Irreducibility and computational equivalence*, pp. 79–88, Springer, 2013.

[12] B. Edmonds, "Complexity and scientific modelling," *Foundations of science*, vol. 5, no. 3, pp. 379–390, 2000.

[13] C. Gershenson and F. Heylighen, "When can we call a system self-organizing?," in *European Conference on Artificial Life*, pp. 606–614, Springer, 2003.

[14] A. WR, "Principles of the self-organizing dynamic system.," *The Journal of General Psychology*, vol. 37, no. 2, pp. 125–128, 1947.

[15] C. Gershenson and N. Fernández, "Complexity and information: Measuring emergence, self-organization, and homeostasis at multiple scales," *Complexity*, vol. 18, no. 2, pp. 29–44, 2012.

[16] R. Fossion, A. L. Rivera, and B. Estañol, "A physicist's view of homeostasis: how time series of continuous monitoring reflect the function of physiological variables in regulatory mechanisms," *Physiological measurement*, vol. 39, no. 8, p. 084007, 2018.

[17] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality," *Physical review A*, vol. 38, no. 1, p. 364, 1988.

[18] M. Suzuki, "Phase transition and fractals," *Progress of Theoretical Physics*, vol. 69, no. 1, pp. 65–76, 1983.

[19] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality: An explanation of the 1/f noise," *Physical review letters*, vol. 59, no. 4, p. 381, 1987.

[20] P. Bak, K. Chen, and M. Creutz, "Self-organized criticality in the'game of life," *Nature*, vol. 342, no. 6251, pp. 780–782, 1989.

[21] R. Vicente Solé, S. Cuevas Manrubia, L. Bartolo, J. Delgado Pin, and J. Bascompte, "Phase transitions and complex systems," 1995.

[22] B. J. Berry, "Cities as systems within systems of cities," *Papers in regional science*, vol. 13, no. 1, pp. 147–163, 1964.

[23] M. Batty, "Building a science of cities," *Cities*, vol. 29, pp. S9–S16, 2012.

[24] J. W. Forrester, "Urban dynamics," *IMR; Industrial Management Review (pre-1986)*, vol. 11, no. 3, p. 67, 1970.

[25] M. Batty, "Modelling cities as dynamic systems," *Nature*, vol. 231, no. 5303, pp. 425–428, 1971.

[26] A. Wilson, *Catastrophe theory and bifurcation: applications to urban and regional systems*. University of California Press, Berkeley, CA, 1981.

[27] C. A. Dawson, *The city as an organism.* University of Toronto Press, 1926.

[28] K. Lynch, *Good city form.* MIT press, 1984.

[29] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework," in *2012 45th Hawaii international conference on system sciences*, pp. 2289–2297, IEEE, 2012.

[30] L. M. Bettencourt, J. Lobo, D. Helbing, C. Kühnert, and G. B. West, "Growth, innovation, scaling, and the pace of life in cities," *Proceedings of the national academy of sciences*, vol. 104, no. 17, pp. 7301–7306, 2007.

[31] M. Kleiber *et al.*, "Body size and metabolism," *Hilgardia*, vol. 6, no. 11, pp. 315–353, 1932.

[32] V. M. Savage, J. F. Gillooly, W. H. Woodruff, G. B. West, A. P. Allen, B. J. Enquist, and J. H. Brown, "The predominance of quarter-power scaling in biology," *Functional Ecology*, vol. 18, no. 2, pp. 257–282, 2004.

[33] G. B. West, J. H. Brown, and B. J. Enquist, "A general model for the origin of allometric scaling laws in biology," *Science*, vol. 276, no. 5309, pp. 122–126, 1997.

[34] J. H. Brown, V. K. Gupta, B.-L. Li, B. T. Milne, C. Restrepo, and G. B. West, "The fractal nature of nature: power laws, ecological complexity and biodiversity," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 357, no. 1421, pp. 619–626, 2002.

[35] M. Batty, "The size, scale, and shape of cities," *science*, vol. 319, no. 5864, pp. 769–771, 2008.

[36] M. Batty and P. A. Longley, *Fractal cities: a geometry of form and function*. Academic press, 1994.

[37] L. Benguigui, D. Czamanski, M. Marinov, and Y. Portugali, "When and where is a city fractal?," *Environment and planning B: Planning and design*, vol. 27, no. 4, pp. 507–519, 2000.

[38] R. Prieto Curiel, L. Pappalardo, L. Gabrielli, and S. R. Bishop, "Gravity and scaling laws of city to city migration," *PloS one*, vol. 13, no. 7, p. e0199892, 2018.

[39] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[40] A. D. Broido and A. Clauset, "Scale-free networks are rare," *Nature Communications*, vol. 10, Mar 2019.

[41] P. Montebruno, R. J. Bennett, C. Van Lieshout, and H. Smith, "A tale of two tails: Do power law and lognormal models fit firm-size distributions in the mid-victorian era?," *Physica A: Statistical Mechanics and its Applications*, vol. 523, pp. 858–875, 2019.

[42] P. Verma and F. Zhang, *The Economics of Telecommunication Services*. Springer, 2020.

[43] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[44] Z. Neal, "Is the urban world small? the evidence for small world structure in urban networks," *Networks and Spatial Economics*, vol. 18, no. 3, pp. 615–631, 2018.

[45] C. Gershenson, "Guiding the self-organization of cyber-physical systems," *Frontiers in Robotics and AI*, vol. 7, p. 41, 2020.

[46] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queueing network control," in *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*, pp. 318–322, IEEE, 1994.

[47] C. Gershenson and D. A. Rosenblueth, "Adaptive self-organization vs static optimization: A qualitative comparison in traffic light coordination," *Kybernetes*, 2012.

[48] S. Lämmer and D. Helbing, "Self-stabilizing decentralized signal control of realistic, saturated network traffic," Citeseer, 2010.

[49] M. A. Turnquist and S. W. Blume, "Evaluating potential effectiveness of headway control strategies for transit systems," *Transportation Research Record*, vol. 746, no. 1, pp. 25–29, 1980.

[50] C. Gershenson and L. A. Pineda, "Why does public transport not arrive on time? the pervasiveness of equal headway instability," *PloS one*, vol. 4, no. 10, p. e7292, 2009.

[51] A. Kesting, M. Treiber, M. Schönhof, and D. Helbing, "Adaptive cruise control design for active congestion avoidance," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 6, pp. 668–683, 2008.

[52] C. Ratti, D. Frenchman, R. M. Pulselli, and S. Williams, "Mobile landscapes: using location data from cell phones for urban analysis," *Environment and planning B: Planning and design*, vol. 33, no. 5, pp. 727–748, 2006.

[53] G. Carreón, C. Gershenson, and L. A. Pineda, "Improving public transportation systems with self-organization: A headway-based model and regulation of passenger alighting and boarding," *PloS one*, vol. 12, no. 12, p. e0190100, 2017.

[54] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig, "Ridesharing: The state-of-the-art and future directions," *Transportation Research Part B: Methodological*, vol. 57, pp. 28–46, 2013.

[55] N. D. Chan and S. A. Shaheen, "Ridesharing in north america: Past, present, and future," *Transport reviews*, vol. 32, no. 1, pp. 93–112, 2012.

[56] N. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang, "Dynamic ride-sharing: A simulation study in metro atlanta," *Procedia-Social and Behavioral Sciences*, vol. 17, pp. 532–550, 2011.

[57] O. Habitat, "Reporte nacional de movilidad urbana en méxico 2014-2015," *Ciudad de México: ONU Habitat*, 2015.

[58] C. E. García Tejeda, "La regulación de uber en la ciudad de méxico, la ganancia de los consumidores y el problema público de la movilidad," *The Latin American and Iberian Journal of Law and Economics*, vol. 2, no. 2, p. 3, 2016.

[59] Instituto Mexicano para la Competitividad, "El peso de los estacionamientos." `https://imco.org.mx/el-peso-de-los-estacionamientos/`, 2016.

[60] Jake Fellows, "Is splitting off resources for your database right for you?." `https://www.liquidweb.com/blog/is-splitting-off-resources-for-your-database-right-for-you/`, 2018.

[61] Unified Compliance Framework, "Application security and development security technical implementation guide, finding-id v-222620." `https://www.stigviewer.com/stig/application_security_and_development/2020-09-30/finding/V-222620`, 2020.

[62] S. Sawlani, *Explaining the Performance of Bidirectional Dijkstra and A* on Road Networks*. PhD thesis, University of Denver, 2017.

[63] M. Schreieck, H. Safetli, S. A. Siddiqui, C. Pflügler, M. Wiesche, and H. Krcmar, "A matching algorithm for dynamic ridesharing," *Transportation Research Procedia*, vol. 19, pp. 272–285, 2016.

[64] E. G. Nilsson, "Design patterns for user interface for mobile applications," *Advances in engineering software*, vol. 40, no. 12, pp. 1318–1328, 2009.

[65] Google Developers, "Design and quality guidelines for android." `https://developer.android.com/design`, 2021.

[66] Apple Inc., "Human interface guidelines for ios." `https://developer.apple.com/design/human-interface-guidelines/`, 2021.

[67] Google Developers, "Build more accesible apps." `https://developer.android.com/guide/topics/ui/accessibility`, 2021.

[68] Apple Inc., "Accessibility for developers." `https://developer.apple.com/accessibility/`, 2021.

[69] L. Punchoojit and N. Hongwarittorrn, "Usability studies on mobile user interface design patterns: a systematic literature review," *Advances in Human-Computer Interaction*, vol. 2017, 2017.

[70] T. Neil, *Mobile design pattern gallery: UI patterns for smartphone apps*. .º'Reilly Media, Inc.", 2014.